625-CD-611-001

EOSDIS Core System Project

ECS Project Training Material
Volume 11: Database Administration

March 2001

Raytheon Systems Company
Upper Marlboro, Maryland

ECS Project Training Material
Volume 11: Database Administration

March 2001

Prepared Under Contract NAS5-60000
CDRL Item 129

RESPONSIBLE ENGINEER

Michael J. Blumenthal Date
EOSDIS Core System Project

SUBMITTED BY

Gary Sloan, M&O Manager Date
EOSDIS Core System Project

Raytheon Systems Company
Upper Marlboro, Maryland

This page intentionally left blank.

Preface

This document is a contract deliverable with an approval code of 3. As such, it does not require formal Government approval. This document is delivered for information only, but is subject to approval as meeting contractual requirements.

Any questions should be addressed to:

Data Management Office
The ECS Project Office
Raytheon Systems Company
1616 McCormick Dr.
Upper Marlboro, MD 20774-5301

Note: This document contains change bars to indicate the addition or revision of material since the issuance of the predecessor document containing training material for Release 6A of the Earth Observing System Data and Information System (EOSDIS) Core System (ECS).

This page intentionally left blank.

Abstract

This is Volume 11 of a series of lessons containing the training material for Release 6A of the Earth Observing System Data and Information System (EOSDIS) Core System (ECS). This lesson provides a detailed description of the process required to perform the tasks associated with database administration.

***Keywords**:* training, database administration, course objective, metadata

This page intentionally left blank.

Change Information Page

List of Effective Pages	
Page Number	**Issue**
Title	Original
iii through xiii	Original
1 through 106	Original
Slide Presentation 1 through 49	Original

Document History			
Document Number	**Status/Issue**	**Publication Date**	**CCR Number**
625-CD-611-001	Original	March 2001	

This page intentionally left blank.

Contents

Overview

Database Devices

User Databases

Database Segments

Database Objects

Transaction Logs

Release 6 Databases

ECS Database Environment

Database Administrator Responsibilities

Database Security and Auditing

Integrity Monitoring

Troubleshooting

ECS Sybase Replication Server Administration Overview

Practical Exercises

Slide Presentation

This page intentionally left blank.

Introduction

Identification

Training Material Volume 11 is part of Contract Data Requirements List (CDRL) Item 129, whose requirements are specified in Data Item Description (DID) 625/OP3 and is a required deliverable under the Earth Observing System Data and Information System (EOSDIS) Core System (ECS), Contract (NAS5-60000).

Scope

Training Material Volume 11 describes the processes and procedures required to accomplish Database Administration. This lesson is designed to provide the operations staff with sufficient knowledge and information to satisfy all lesson objectives.

Purpose

The purpose of this lesson is to provide a detailed course of instruction that forms the basis for understanding Database Administration. Lesson objectives are developed and will be used to guide the flow of instruction for this lesson. The lesson objectives will serve as the basis for verifying that all lesson topics are contained within this Student Guide and slide presentation material.

Status and Schedule

This lesson module provides detailed information about training for Release 6A. Subsequent revisions will be submitted as needed.

Organization

This document is organized as follows:

Introduction:	The Introduction presents the document identification, scope, purpose, and organization.
Related Documentation:	Related Documentation identifies parent, applicable and information documents associated with this document.
Student Guide:	The Student Guide identifies the core elements of this lesson. All Lesson Objectives and associated topics is included.
Slide Presentation:	Slide Presentation is reserved for all slides used by the instructor during the presentation of this lesson.

This page intentionally left blank.

Related Documentation

Parent Document

The parent document is the document from which this ECS Training Material's scope and content are derived.

423-41-01	Goddard Space Flight Center, EOSDIS Core System (ECS) Statement of Work

Applicable Documents

The following documents are referenced within this ECS Training Material, or are directly applicable, or contain policies or other directive matters that are binding upon the content of this document:

420-05-03	Goddard Space Flight Center, Earth Observing System (EOS) Performance Assurance Requirements for the EOSDIS Core System (ECS)
423-41-02	Goddard Space Flight Center, Functional and Performance Requirements Specification for the Earth Observing System Data and Information System (EOSDIS) Core System (ECS)

Information Documents

Information Documents Referenced

The following documents are referenced herein and amplify or clarify the information presented in this document. These documents are not binding on the content of the ECS Training Material.

609-CD-600	Release 6A Operations Tools Manual for the ECS Project
611-CD-600	Mission Operation Procedures for the ECS Project
910-TDA-022	Custom Configuration Parameters for ECS Release 6A

Information Documents Not Referenced

The following documents, although not referenced herein and/or not directly applicable, do amplify or clarify the information presented in this document. These documents are not binding on the content of the ECS Training Material.

305-CD-600	Release 6A Segment/Design Specification for the ECS Project
311-CD-600	Release 6A Data Management Subsystem Database Design and Database Schema Specifications for the ECS Project
311-CD-601	Release 6A Ingest Database Design and Database Schema Specifications for the ECS Project

311-CD-602	Release 6A Interoperability Subsystem (IOS) Database Design and Database Schema Specifications for the ECS Project
311-CD-603	Release 6A Planning and Data Processing Subsystem Database Design and Schema Specifications for the ECS Project
311-CD-604	Release 6A Science Data Server Database Design and Schema Specifications for the ECS Project
311-CD-605	Release 6A Storage Management and Data Distribution Subsystems Database Design and Database Schema Specifications for the ECS Project
311-CD-606	Release 6A Subscription Server Database Design and Schema Specifications for the ECS Project
311-CD-607	Release 6A Systems Management Subsystem Database Design and Schema Specifications for the ECS Project
311-CD-608	Release 6A Registry Database Design and Schema Specifications for the ECS Project
313-CD-600	Release 6A ECS Internal Interface Control Document for the ECS Project
334-CD-600	6A Science System Release Plan for the ECS Project
601-CD-001	Maintenance and Operations Management Plan for the ECS Project
603-CD-003	ECS Operational Readiness Plan for Release 2.0
604-CD-001	Operations Concept for the ECS Project: Part 1-- ECS Overview
604-CD-002	Operations Concept for the ECS Project: Part 2B -- ECS Release B
605-CD-002	Release B SDPS/CSMS Operations Scenarios for the ECS Project
607-CD-001	ECS Maintenance and Operations Position Descriptions
152-TP-001	ACRONYMS for the EOSDIS Core System (ECS) Project
152-TP-003	Glossary of Terms for the EOSDIS Core System (ECS) Project
211-TP-005	Transition Plan 4PX to 4PY, 4PY to 5A, and 5A to 5B for the ECS Project
220-TP-001	Operations Scenarios - ECS Release B.0 Impacts
500-1002	Goddard Space Flight Center, Network and Mission Operations Support (NMOS) Certification Program, 1/90
535-TIP-CPT-001	Goddard Space Flight Center, Mission Operations and Data Systems Directorate (MO&DSD) Technical Information Program Networks Technical Training Facility, Contractor-Provided Training Specification

Database Administration

Lesson Overview

This lesson will provide you with the tools needed to perform the various tasks required to administer and maintain the database and structure management for the Earth Observing System Data and Information System (EOSDIS) Core System (ECS) during maintenance and operations.

Lesson Objectives

Overall Objective - This lesson provides a detailed description of the different tasks required to maintain the database and structure management for ECS, provide the operations interface to perform database administration utilities such as product installation and disk storage management, managing user accounts and privileges, backup and recovery, monitoring physical allocation of database resource information, loading metadata and maintaining metadata.

Condition - The student will be given a copy of *625-CD-611-001 ECS Project Training Material Volume 11 Database Administration* and a functioning system.

Standard - The student will perform without error the procedures required to perform database administration.

Specific Objective 1 - The student will be able to create new database devices, allocate appropriate disk space to house the new database, and maintain database segments including managing and monitoring the use of available disk space, memory, connection error logs, state of transaction logs, device problems, etc.

Condition - The student will be given a copy of *625-CD-611-001 ECS Project Training Material Volume 11: Database Administration* and a functioning system.

Standard - The student will perform without error the procedures relating to the creation of new database devices, the allocation of appropriate disk space, and maintenance of database segments.

Specific Objective 2 - The student will able to start and shutdown the SQL server.

Condition - The student will be given a copy of *625-CD-611-001 ECS Project Training Material Volume 11: Database Administration* and a functioning system.

Standard - The student will perform without error the procedures relating the startup and shutdown of the SQL server.

Specific Objective 3 - The student will be able to perform database user account and access privilege procedures including:

- Creating user accounts.
- Granting and revoking access privileges for data retrieval, insertion, deletion and update of objects.

- Granting and revoking roles for SQL server users groups.

Condition - The student will be given a copy of *625-CD-611-001 ECS Project Training Material Volume 11: Database Administration* and a functioning system.

Standard - The student will perform without error the procedures relating to the creation of user accounts and the granting and revoking of access privileges.

Specific Objective 4 - The student will be able to perform database security and auditing procedures.

Condition - The student will be given a copy of *625-CD-611-001 ECS Project Training Material Volume 11: Database Administration* and a functioning system.

Standard - The student will perform without error the procedures relating to database security and auditing procedures.

Specific Objective 5 - The student will be able to perform database integrity monitoring.

Condition - The student will be given a copy of *625-CD-611-001 ECS Project Training Material Volume 11: Database Administration* and a functioning system.

Standard - The student will perform without error the procedures relating to database integrity monitoring.

Specific Objective 6 - The student will be able to perform database backups on a regular or on-demand basis.

Condition - The student will be given a copy of *625-CD-611-001 ECS Project Training Material Volume 11: Database Administration* and a functioning system.

Standard - The student will perform without error the procedures relating to database backups.

Specific Objective 7 - The student will be able to perform a recovery of the database following a system failure or on-demand.

Condition - The student will be given a copy of *625-CD-611-001 ECS Project Training Material Volume 11: Database Administration* and a functioning system.

Standard - The student will perform without error the procedures relating to database recovery.

Specific Objective 8 - The student will be able to configure databases unique to ECS DAACs including:

- Making database size estimates and planning.
- Preparing database-unique attributes.
- Preparing database reports.

Condition - The student will be given a copy of *625-CD-611-001 ECS Project Training Material Volume 11: Database Administration* and a functioning system.

Standard - The student will perform without error the procedures relating to database configuration specific to the ECS DAACs.

Specific Objective 9 - The student will be able to perform database tuning and performance monitoring procedures including:

- Design and indexing.
- Responding to queries.
- Monitoring and boosting performance.

Condition - The student will be given a copy of *625-CD-611-001 ECS Project Training Material Volume 11: Database Administration* and a functioning system.

Standard - The student will perform without error the procedures relating to database tuning and performance monitoring.

Specific Objective 10 - The student will able to describe Sybase Replication Server Administration.

Condition - The student will be given a copy of *625-CD-611-001 ECS Project Training Material Volume 11: Database Administration* and a functioning system.

Standard - The student will be able to describe Sybase Replication Server Administration.

Importance

ECS relies on vast amounts of data from the science user perspective and from a maintenance and operations perspective. Accurate information stored in the science databases allows science user's to access necessary data quickly. Similarly, databases that maintain operational data must be kept current in order for routine and specialized administrative tasks to be performed.

Special Instructions on Procedures

All procedures in this training guide assume that you are logged in to a system server or workstation as yourself.

Many DBA tasks will be performed by using scripts that are run from the command line. Some scripts may require modification which will require you to use a text editor of your choice. These procedures do not give keystroke-by-keystroke information on how to edit the file; they only instruct you to make the changes required to perform the task.

This page intentionally left blank.

Overview

Overview of Database Administration

The Database Administrator or DBA, is the individual or group responsible for the installation, configuration, update, maintenance, and overall integrity, performance and reliability of the SQL Server database. In general, the DBA is concerned with the availability of the server, the definition and management of resources allocated to the server, the definition and management of databases and objects resident on the server, and the relationship between the server and the operating system.

The terms described in the following table will be used throughout this chapter.

Table 1. SQL Server General Definitions.

Term	Definition
SQL Server	The server in the Sybase client/server architecture. SQL Server manages multiple databases and multiple users, keeps track of the actual location of data on disks, maintains mapping of logical data description to physical data storage, and maintains data and procedure caches in memory.
Client	SYBASE Open Client software located in the /tools/sybOCv(TBD) directory for SUN and HP platforms SYBASE Open Client software located in the /tools/sybOCv(TBD) directory for SGI platform
Backup Server	Similar to the dataserver, it uses a separate UNIX process to off load the cycles associated with DUMP and LOAD commands
backups	The set of UNIX files containing full database dumps, transaction log dumps, and dbcc output
dbcc	Database Consistency Checker - a utility program designed to check the logical and physical consistency of a database
sybase root directory	/usr/ecs/OPS/COTS/sybase, this is the home directory for all SYBASE software and related products and is referenced both in UNIX and in the rest of this document as **$SYBASE**
interfaces file	Lists the names and access paths for all servers and backup servers. This file is located in the **$SYBASE**
sa	System Administrator login, this is the superuser of the SQL Server
scripts	UNIX script programs located in **$SYBASE**/scripts and related subdirectories {$ecs_Home}/{mode}/custom/dbms/{subsystem}
showserver	A utility invoked at the UNIX command prompt to display active servers, located in **$SYBASE**/install.
SQL scripts	SQL and command statements located in **$SYBASE**/scripts and related subdirectories and /{$ecs_Home}/{mode}/custom/dbms/{subsystem}
Server Name	The name of the database server for a specific application in different modes **EX. -** PDPS application database server in OPS mode EX.- Pdps_TS1 in TS1 mode EX. -pdps _TS2 in TS2 mode

Port Numbers	The port number to be utilized by the above listed servers.
Release Directory	$SYBASE
SQL	Structured Query Language

Database Devices

A database device stores the objects that make up databases. The term *device* does not necessarily refer to a distinct physical device; it can refer to any piece of disk, such as a partition; or a file in the file system used to store databases and their objects (Figure 1).

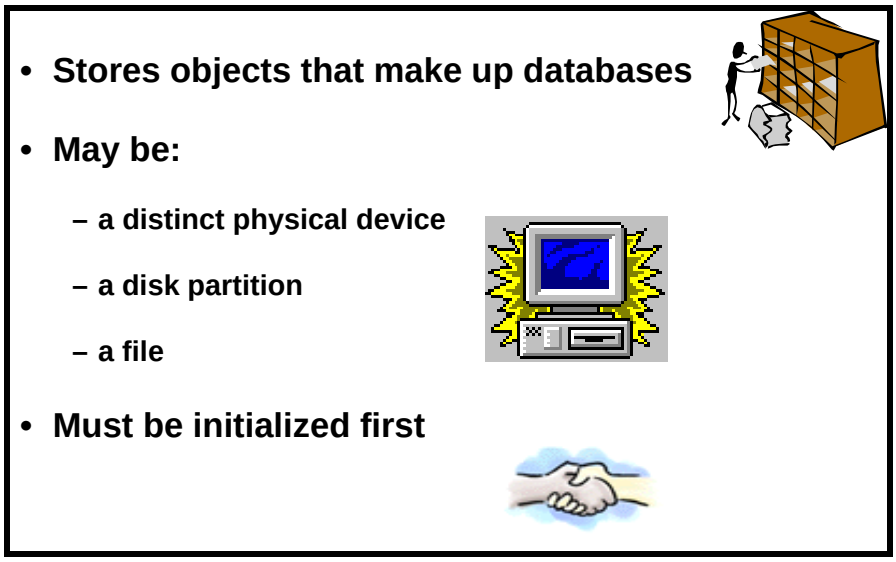


Figure 1. Database Devices.

Each database device or file must be prepared and made known to the SQL Server before it can be used for database storage, the process is called initialization. Once a database device is initialized, it can be:

- Allocated to the pool of space available to a user database.
- Allocated to a user database, and assigned to store a specific database object or objects.
- Used to store a database's transaction logs.
- Designated as a default device for **create** and **alter** database commands.

A database device is created when the System Administrator determines that new disk space is available for use by a Sybase database, or as part of the Database Recovery Procedure. The System Administrator makes a request to the DBA who creates the new database device and notifies the System Administrator when the device has been created.

In order to perform the procedure, the DBA must have obtained the following information:

- Name of database device to create.
- Physical device on which to place database device.
- Device size in megabytes.
- For a mirrored device, name of the mirror device (at this printing, no mirroring is in effect).

Making Database Size Estimates and Planning

Before a database is created, a user has to estimate the size of the database required based on how much data is expected to be stored. This estimate will be the database size. By default, the log size will be $1/5^{th}$ the data size. While estimating the database size, a user has to keep in mind the future growth.

Creating Logical Devices

A logical device is created when the UNIX System Administrator determines that new disk space is available for use by SYBASE software, databases, transaction logs, and/or backups. Either raw disk partitions or UNIX filesystem partitions can be used to create a logical device. The creation of a logical device is a mapping of physical space to a logical name and virtual device number (**vdevno**) contained in the SQL Server **master** database. The **disk init** command is used to initialize this space. After the disk initialization is complete, the space described by the physical address is available to SQL Server for storage, and a row is added to the **sysdevices** table in the **master** database.

A System Administrator initializes new database devices with the disk init command.

Disk Init does the following: Maps the specified physical disk device or operating system file to a database device name

Lists the new device in master..sysdevices

Prepares the device for database storage

Note

Before you run disk init, see the SQL Server installation and configuration guide for your platform for information about choosing a database device and preparing it for use with SQL Server. You may want to repartition the disks on your computer to provide maximum performance for your Sybase databases.

Disk init divides the database devices into allocation units of 256 2K pages, a total of 1/2MB. In each 256-page allocation unit, the disk init command initializes the first page as the allocation page, which will contain information about the database (if any) that resides on the allocation unit.

Note

After you run the disk init command, be sure to use dump database to dump the master database. This makes recovery easier and safer in case master is damaged. If you add a device and fail to back up master, you may be able to recover the changes with disk reinit.

Syntax: disk init

name = "device_name" ,

physname = "physicalname" ,

vdevno = virtual_device_number ,

size = number_of_blocks

[, vstart = virtual_address ,

cntrltype = controller_number]

Example of Creating a Database Device

1 Log on to a local host.

2 Set display to current terminal by typing: **setenv DISPLAY *IPNumber*:0.0** or **setenv DISPLAY *ServerName*:0.0**, then press **Return**.

3 Log into the the server on which the new database device is to be created by typing: **/tools/bin/ssh <*ServerName*>**, then press **Return**.

- If you have previously set up a secure shell passphrase and executed **sshremote**, a prompt to **Enter passphrase for RSA key '<*user@localhost*>'** appears; continue with Step 4.
- If you have not previously set up a secure shell passphrase; go to Step 5.

4 If a prompt to **Enter passphrase for RSA key '<*user@localhost*>'** appears, type your ***Passphrase*** and then press the **Enter** key. Go to step 6.

5 At the **<*user@remotehost*>'s password:** prompt, type your ***Password*** and then press the **Enter** key.

For each database device to be created, perform Steps 6 through 9:.

6 At the UNIX prompt, type **cd server.devices**, then press **Return.**

- This places you in the directory that contains all the scripts used to create database devices.
- It is important that these scripts not be deleted from the system in case it is necessary to restore a database device following a failure (see *Restore Database Devices*).

7 At the UNIX prompt, type **cp template.sql *DeviceName*.sql**, then press **Return**.

- This creates a new SQL script file into which you will type the appropriate commands to create the database device.

8 Using the text editor of your choice, edit ***DeviceName*.sql** (Figure 2) and make changes to information enclosed in brackets (including the brackets) as appropriate:

```
/*****************************************************/
/* name: [add_devices.sql]                           */
/* purpose:                                          */
/* written:                                          */
/* revised:                                          */
/* reason:                                           */
/*****************************************************/
disk init name = [device name],
physname = "/dev/[device name]",
vdevno = [#],
size = [size]
go
sp_helpdevice [device name]
go
```

Figure 2. add_devices.sql file for creation of a database device.

- In the area delimited by **/* */**, enter an appropriate description of the script including the file name, date written and person who wrote the script, its purpose, and any other information deemed appropriate. Be sure to enclose each line of the comment between **/* */**.
- The **disk init name** is the ***DeviceName*** is used in Step 8 above.
  - You may not use spaces or punctuation except the underscore character (_) in ***DeviceName***.
  - Remember that the name you assign is case sensitive.
  - Be sure there is a comma after ***DeviceName***
- The **physname** is the ***FullPath_to_DeviceName***..
  - Be sure to enclose ***FullPath_to_DeviceName*** in double quotes.
  - Be sure to place a comma after ***FullPath_to_DeviceName*** but outside the double quotes.

- The **vdevno** is the ***VirtualDeviceNumber***
  - **vdevno** is a unique identifying number for the database device. It must be unique among all the devices used by SQL Server and is never reused. Device number 0 represents the device named **d_master** that stores the system catalogs. Legal numbers are between 1 and 255, but the highest number must be one less than the number of database devices for which your system is configured. For example, for a system with the default configuration of 10 devices, the legal device numbers are 1-9. The default of 10 devices can be changed using **sp_configure**. For the GDAAC implementation, the next vdevno is located in the /usr/dba0x/server.devices/next.vdevno file, it should be incremented when used. If this file does not exist, the next available number can be determined by looking at the output from the **sp_helpdevice** command and selecting the next number in sequence. If you use an existing number, Sybase will return the message, "device activation error."
- The **size** is the ***DeviceSize*** in blocks.
  - To compute the number of blocks, multiply the device size in megabytes by 512; e.g., a 1,000 Mb device has 512,000 blocks

9 After the changes have been made, save the file according to the rules of your text editor.

10 At the UNIX prompt, type:

isql –U<username> -S<ServerName -i*DeviceName*.sql -o*Device Name.out*

then press **Return**.

- ***ServerName*** is the name of the server on which the database device will be created.
- ***DeviceName.sql*** is the name of the script file you created in step 8.
- ***DeviceName.out*** is the filename of the script's output for confirmation and/or troubleshooting purposes.
- The system will prompt you for a password.

11 At the **Password:** prompt, type the **<*password*>**, then press **Return.**

12 When the UNIX prompt is again displayed the process is complete.

13 At the UNIX prompt, type **more *DeviceName.out***, then press **Return**.

- This allows you to view the ***DeviceName.out*** file to confirm that the device has been created or to check for device creation errors.

A sample of a completed Database Device creation script follows (Figure 3).

```
/*******************************************************/
/* name: test_dev.sql                                  */
/* purpose: allocate 3Mb device for testing            */
/* written: 12/18/97                                   */
/* revised:                                            */
/* reason:                                             */
/*******************************************************/
disk init name = test_dev,
physname =
     “/usr/ecs/Rel_A/COTS/sybase/studentdevices/test_dev.dat”,
vdevno = 15,
size = 1536
go
sp_helpdevice test_dev
go
```

Figure 3. Completed database device creation script.

User Databases

Creating and Altering Databases

User Databases

A user database is created when an approved request is received by the DBA, or as part of the Database Recovery Procedure (see **Database Recovery)**. The database name, approximate size of the database and the transaction log are necessary information for the DBA to complete the task of creating a user database. It is the user's responsibility to determine the appropriate database design.

The requester fills out a "User Database Request Form" and submits it to the supervisor.

The requester's supervisor reviews the request, and if it's determines that it is appropriate to create the User Database, forwards the request to the Operations Supervisor (Ops Super). The Ops Super verifies that all required information is contained on the form. (Incomplete forms are returned to the requester's supervisor for additional information.) If it is complete and if the request for a new User Database fits within policy guidelines, the Ops Super approves the request and forwards the request form to the DBA to implement. After the User Database is created, the DBA notifies the requester that the new database is available. The DBA also sends an e-mail message to the user's supervisor informing them that the new User Database was created.

In order to perform the procedure, the DBA must have obtained the following information:

- Name of database to create.
- Size of database and the transaction log.
- Database devices to use (some may need to be created).
- To create a database, the DBA must have sa_role (SQL Server) privileges.

A user database is created by the DBA with a script containing the **create database** command. A database is created on one or more physical devices. Specifying the device is optional - but highly recommended. When indicating the device, you use the logical name you specified as part of a **disk init** (described above). Unlike the **disk init** command, the size of the database data and log components is specified in MB instead of 2K pages.

Example of Creating a User Database

1 Log on to a local host.

2 Set display to current terminal by typing: **setenv DISPLAY *IPNumber*:0.0** or **setenv DISPLAY *ServerName*:0.0**, then press **Return**.

3 Log into the the server on which the new database device is to be created by typing: **/tools/bin/ssh <*ServerName*>**, then press **Return**.

- If you have previously set up a secure shell passphrase and executed **sshremote**, a prompt to **Enter passphrase for RSA key '<*user@localhost*>'** appears; continue with Step 4.
- If you have not previously set up a secure shell passphrase; go to Step 5.

4 If a prompt to **Enter passphrase for RSA key '<*user@localhost*>'** appears, type your ***Passphrase*** and then press the **Enter** key. Go to step 6.

5 At the **<*user@remotehost*>'s password:** prompt, type your ***Password*** and then press the **Enter** key.

For each database device to be created, perform Steps 6 through 9:

6 At the UNIX prompt, type **cd server.databases**, then press **Return.**

- This places you in the directory that contains all the scripts used to create database devices.
- It is important that these scripts not be deleted from the system in case it is necessary to restore a database device following a failure (see **Restore Databases**).

7 At the UNIX prompt, type **cp template.sql *DBName*.sql**, then press **Return**.

- This creates a new SQL script file into which you will type the appropriate commands to create the user database.

8 Using the text editor of your choice, edit ***DBName*.sql** (Figure 5) and make changes to information enclosed in brackets (including the brackets) as appropriate:

- **DBName** is the name of the database and should describe its function succinctly.
- **DBDeviceName** is the name of the existing, approved database device on which **DBName** will reside.
- **DBSize_in_MB** is the estimated size of the database in megabytes.
- **LogDBDeviceName** is the name of the database device holding the transaction log
- **LogSize_in_MB** is the estimated size of the transaction log in megabytes.

- The **sp_helpdb** command provides feedback at the end of the script to assure that the database has been created.

```
/* ********************************************** */
/* name: template.sql                             */
/* purpose:                                       */
/* written:                                       */
/* revised:                                       */
/* reason:                                        */
/* ********************************************** */
create database database_name
on data_dev = size, /* in MB */
log on log_dev = size
go
sp_helpdb database_name
go
```

Figure 4. Sample template.sql file for creation of a database.

9 After the changes have been made, save the file according to the rules of the text editor.

10 At the UNIX prompt, type:

isql –U<username> -S<ServerName> -i*DBName*.sql -o*DBName*.out

then press **Return**.

- ***ServerName*** is the name of the server on which the database device will be created.
- ***DBName.sql*** is the name of the script file you created in step 8.
- ***DBName.out*** is the filename of the script's output for confirmation and/or troubleshooting purposes.
- The system will prompt you for a password.

11 At the **Password:** prompt, type the **<*password*>**, then press **Return.**

12 When the UNIX prompt is again displayed the process is complete.

13 At the UNIX prompt, type **more *DBName.out*** , then press **Return**.

- This allows you to view the ***DBName.out*** file to confirm that the device has been created or to check for database creation errors.

A sample of a completed Create Database script follows (Figure 5).

```
/*****************************************/
/* name: test_db.sql                     */
/* purpose: create a database for testing */
/* written: 12/19/97                     */
/* revised:                              */
/* reason:                               */
/*****************************************/
create database test_db
on test_dev = 3
go
sp_helpdb test_db
go
```

Figure 5. Completed Create DatabaseScript.

Example of Altering a Database

The user database **UserDB** has run out of space and it has been determined that it should be increased by 50MB.

1 In **$SYBASE**/scripts/create.databases, DBA creates a script file containing the ALTER DATABASE command (named alter_userdb.sql)

Syntax: Alter database UserDB on data_dev3 = 50

2 DBA runs the script from the UNIX command prompt:

Syntax: % isql -Usa -S**servername** -ialter_userdb.sql -oalter_userdb.out

3 DBA checks the alter_userdb.out file for success

Database Segments

Data Placement - Segmentation

Segments are named subsets of the database devices available to a particular SQL Server database. Segment names are used in **create table** and **create index** commands to place tables or indexes on specific database devices. Using segments allows the DBA to better control the size of database objects and may improve performance by spreading i/o more evenly across devices.

Once the database device exists and is available, the segment can be defined with the system stored procedure **sp_addsegment**.

Syntax: sp_addsegment segname, dbname, devname

After the segment has been defined in the current database, the **create table** or **create index** commands use the optional clause "on segment_name" to place the object on a particular segment.

Syntax: create table table_name (column_name datatype ...) [on segment_name]

create [clustered | nonclustered] index index_name on table_name (columns)

Use **sp_helpdb** database_name to display the segments defined for that database.

Use **sp_helpsegment** segment_name to list the objects on the segment and show the mapped devices.

Creating Segments

A **segment** is a collection of the database devices and/or fragments available to a particular database. Tables and indexes can be assigned to a particular segment, and thereby to a particular physical device, or can span a set of physical devices.

Segments are named subsets of the database devices available to a particular SQL Server database. A segment can best be described as a label that points to one or more database devices. Segment names are used in **create table** and **create index** commands to place tables or indexes on specific database devices. The use of segments can increase SQL Server performance, and can give the System Administrator or Database Owner increased control over placement, size and space usage of specific database objects. For example:

- If a table is placed on one device, and its non-clustered indexes on a device on another disk controller, the time required to read or write to the disk can be reduced, since disk head travel is usually reduced.
- If a large, heavily-used table is split across devices on two separate disk controllers, read/write time may be improved.

SQL Server stores the data for text and image columns on a separate chain of data pages. By default, this text chain is placed on the same segment as the table. Since reading a text column requires a read operation for the text pointer in the base table and an additional read operation on the text page in the separate text chain, placing the text chain on a separate physical device can improve performance.

If you place tables and indexes only on specific segments, those database objects cannot grow beyond the space available to the devices represented by those segments, and other objects cannot contend for space with them.

Segments can be extended to include additional devices as needed.

You can use thresholds to warn you when space becomes low on a particular database segment.

Segments are created within a particular database from the database devices already allocated to that database. Each SQL Server database can contain up to 32 segments. The database devices must first be initialized with **disk init**, and then be made available to the database with a **create database** or **alter database** statement before you can assign segment names.

When you first create a database, SQL Server creates three segments in the database (Table 2):

Table 2. Default segment names and functions.

Segment	**Function**
system	Stores this database's system tables.
logsegment	Stores this database's transaction log.
default	Stores all other database objects-unless you create additional segments, and store the table or index on the new segments by using **create table... on *segment_name*** or **create index... on *segment_name*.**

Database segments are created when a database is created, when the Database Administrator determines that it is necessary to allocate database objects to new or additional devices, or as part of the Database Recovery Procedure.

In order to perform the procedure, the DBA must have obtained the following information:

- Name of database
- Database device extents
- Space on the database allocated to the Database device

To create a database, the DBA must have sa_role.

Create Database Segment Scenario and Procedure

The DBA receives a request to create a segment for the storage of the SubServer table indexes in the t1ins01_srvr database, on a separate physical disk. Two devices **subserver_data** and **subserver_index** have already been created and are located on different physical disks.

1 At the UNIX prompt, type **cd scripts**, then press **Return**.

2 Using the text editor of your choice, edit ***segment*.sql** (Figure 6) and make changes to information enclosed in brackets (including the brackets) as appropriate:

Note: This file doesn't exist. It must be created.

```
/*****************************************/
/* name: [segment.sql]                   */
/* purpose:                              */
/* written:                              */
/* revised:                              */
/* reason:                               */
/*****************************************/

 sp_addsegment [seg_name], [DBname],[Device Name]
```

Figure 6. Create database segment template file.

3 After the changes have been made, save the file according to the rules of the text editor.

4 At the UNIX prompt, type:**isql –U<username> -S<*ServerName*> -i*Segment*.sql -o*Segment*.out**

then press **Return.**

- The system will prompt you for a password.

5 At the **Password:** prompt, type the **<*password*>**, then press **Return.**

6 When the UNIX prompt is again displayed the process is complete.

7 At the UNIX prompt, type **more *Segment.out*** , then press **Return**.

- This allows you to view the ***Segment.out*** file to confirm that the device has been created or to check for database creation errors.

This page intentionally left blank.

Database Objects

Creating Database Objects

For special cases, creation (and modification) scripts are stored in **$SYBASE**/scripts/scriptname. There should be a template for each type of object to be created.

Objects supported in Sybase SQL Server 11

SQL Server supports many database objects enabling users to utilize, access, and care for back-end data. The following objects are supported in Sybase SQL Server 11:

- Tables, for storage of SQL Server data
- Temporary tables, to store intermediate "set" results
- Views, which provide a logical depiction of data from table(s)
- Rules, that validate column data
- Defaults, to provide a value in a column when none is supplied
- Constraints, which validate column data and ensure consistency with other columns in other tables
- User-defined datatypes
- Indexes, to provide faster access and optionally assure uniqueness
- Keys, which document structure within and between tables
- Triggers, to provide limited event processing
- Stored Procedure, which provide for more complex processing on the back-end (i.e. triggers)

Example of Creating a User Database Table

1 Log on to a local host.

2 Set display to current terminal by typing: **setenv DISPLAY *IPNumber*:0.0** or **setenv DISPLAY *ServerName*:0.0**, then press **Return**.

3 Log into the the server on which the new tabel is to be created by typing: **/tools/bin/ssh <*ServerName*>**, then press **Return**.

- If you have previously set up a secure shell passphrase and executed **sshremote**, a prompt to **Enter passphrase for RSA key '<*user@localhost*>'** appears; continue with Step 4.
- If you have not previously set up a secure shell passphrase; go to Step 5.

4 If a prompt to **Enter passphrase for RSA key '<*user@localhost*>'** appears, type your ***Passphrase*** and then press the **Enter** key. Go to step 6.

5 At the **<*user@remotehost*>'s password:** prompt, type your ***Password*** and then press the **Enter** key.

6 At the UNIX prompt, type **cd database.objects**, then press **Return.**

- This places you in the directory that contains all the scripts used to create database devices.
- It is important that these scripts not be deleted from the system in case it is necessary to restore a database device following a failure (see **Restore Databases**).

7 At the UNIX prompt, type **cp template.sql *TableName*.sql**, then press **Return**.

- This creates a new SQL script file into which you will type the appropriate commands to create the user database table.

8 Using the text editor of your choice, edit ***TableName*.sql** (Figure 7) and make changes to information enclosed in brackets (including the brackets) as appropriate:

- **TableName** is the name of the database table and should describe it's function succinctly.

The **sp_help Table_name** command provides feedback at the end of the script to assure that the database table has been created.

```
/*****************************************/
/* name: [table name].sql                */
/* purpose:                              */
/* written:                              */
/* revised:                              */
/* reason:                               */
/*****************************************/
use [database name]
go
create table [table name] (
[column 1] [datatype] [null | not null]
[column 2] [datatype] [null | not null]
)
go
sp_help [table name]
go
/* for any other objects consult document.*/
```

Figure 7. Sample template.sql file for creation of a database table.

9 After the changes have been made, save the file according to the rules of the text editor.

10 At the UNIX prompt, type:

isql –U<username> -S<ServerName> -i*TableName*.sql -o*Table Name*.out

then press **Return**.

- ***ServerName*** is the name of the server on which the database device will be created.
- ***TableName.sql*** is the name of the script file you created in step 9.
- ***TableName.out*** is the filename of the script's output for confirmation and/or troubleshooting purposes.
- The system will prompt you for a password.

11 At the **Password:** prompt, type the <***password***>, then press **Return.**

12 When the UNIX prompt is again displayed the process is complete.

13 At the UNIX prompt, type **more *TableName.out*** , then press **Return**.

- This allows you to view the ***TableName.out*** file to confirm that the device has been created or to check for database creation errors.

A sample of a completed Create Database Table script follows (Figure 8).

```
/*****************************************/
/* name: test_table.sql                  */
/*  purpose:  create  a  database  table  for     */
     /* testing                                  */
/* written: 12/19/97                     */
/* revised:                              */
/* reason:                               */
/*****************************************/
use [database name]
go
create table test_table   (
last_name varchar(30) not null,
first_name varchar(30) not null,
)
go
sp_helpdb test_table
go
```

Figure 8. Completed Create Table Script.

Transaction Logs

What is a Transaction?

A **transaction** is the SQL Server's basic unit of work. Each SQL statement that modifies data in a database is considered a transaction, and SQL Server uses them to keep track of all database changes. A transaction consists of one or more Transact-SQL statements that succeed or fail as a unit. Each success or failure is recorded automatically in the database's transaction log.

What is a Transaction Log?

The Transaction Log is an ever-expanding file that records each and every transaction that occurs in a database. This log is used to perform a complete recovery of the database in the event of a media failure. The transaction log is normally maintained on a different database device or even a completely separate system than the rest of the database so that it can be read should a failure occur.

How Transaction Logging Works

The transaction log is a write-ahead log. When a user issues a statement that would modify the database, SQL Server automatically opens the transaction log and writes a **transaction start** statement. After all changes for a statement have been recorded, a **transaction end** statement is written to the log, and the log is written to an in-cache copy of the data page. The data page remains in cache until the memory is needed for another database page at which time the **checkpoint** process writes the changes to disk.

If any statement in a transaction fails to complete due to operator error, the operator aborting the procedure, or mechanical failure, SQL Server automatically reverses all changes made by the transaction. SQL Server writes an "end transaction" record to the log at the end of each transaction, recording the status (success or failure) of the transaction.

Each database has its own transaction log that records all changes to its database which may or may not be on the same database device. It is strongly recommended that transaction logs be kept on a device separate from the database as insurance against catastrophic data loss.

The transaction log grows in size over time and may cause database problems when full. Therefore, the transaction log must be dumped periodically.

When modifying data, the server takes the following steps:

1. Writes a *begin* tran record in the log (in cache).
2. Records the modification in the log (in cache).
3. Performs the modification to the data (in cache).

4. Writes a *commit* tran record to the log (in cache).
5. Flushes all “dirty” (modified) log pages to disk.

Getting Information about the Transaction Log

The transaction log contains enough information about each transaction to ensure that it can be recovered. Use the **dump transaction** command to copy the information it contains to tape or disk. Use **sp_spaceused syslogs** to check the size of the log, or **sp_helpsegment logsegment** to check the space available for log growth.

Release 6 Databases

Release 6 Databases

Release 6A databases are divided into two categories:

- Production databases contain data used in the generation of end-user products such as the Earth Science Data Types (ESDTs), Product Generation Executables (PGEs), and metadata.
- System Management databases hold results of system activity and access, such as configuration management, network activity, and trouble tickets.

Table 3 and Table 4 detail the information stored in each database.

Table 3. Release 6A Production Databases

Database	Content
Ingest DB	File types to be ingested, status of ingest production requests, historical summary.
Planning & Data Processing System (PDPS)	PGE information to plan production.
Metadata	Collection, granule, document, producer, and other descriptive information about science data.
MSS Management DB	System and application performance information, user profiles, and order tracking information.
Storage Management Pull Monitor	Staging disk usage, tape drives available, requests needing resources.
Data Dictionary	User descriptions of attributes and other terms in system
Document Database (WEB-based search only)	ECS-related science documents.
ASTER Lookup Table (LUT)	Atmospheric correction parameters
CSS Persistent Access Control List (ACL)	List of principals who can access components.
Advertising Database	Holdings and services available to users.
Subscription Database	Principals (users or servers) to be notified when a specified event occurs.

Table 4. Release 6A System Management Databases

Application	Tool	Server
Network Management	HP OpenView	MSS
System Performance Management	Tivoli TME/Sentry	MSS
Extensible SNMP Agent	Peer Networks Optima	All platforms
Trouble Ticketing	Remedy Corporation ARS	MSS
Physical Configuration Management	Accugraph Corporation (PNM)	MSS
Security/DCE Management	HAL DCE Cell Manager	CSS
Software Change Management	Clearcase	CM
Change Request Management	DDTS	CM
Baseline Manager	HTG XRP	CM

Table 5 below provides a listing of Release 6A SQL Servers.

Table 5. Release 6A SQL Servers at GSFC.

Database	Content
g0acg01_srvr	Data Subsystem Server (Science Dataserver and Storage Management)
g0acg01_backup	
g0ins01_srvr	Subscription Server
g0ins01_backup	
g0mss21_srvr	Management Subsystem Server (ACL and MSS Databases)
g0mss21_backup	
g0ins02_srvr	Data Management Server (Advertising and Data Dictionary)
g0ins02_backup	
g0pls02_srvr	Planning and Data Processing Server (Autosys)
g0pls02_backup	
g0icg01_srvr	Ingest Server
g0icg01_backup	

Naming Conventions

In order to keep track of the large number of files that you will create and work with throughout your tenure as DBA, and because other people will come into contact with these files at one time or another, please follow these simple naming conventions for your files:

- The file name should indicate the function and/or content of the object regardless of the length of the file name.
- Only easily understandable abbreviations should be used.
- Parts of names are separated by the underscore character (_).
- An maximum of one suffix is permitted and is appended to the file name by a period (.).
- The full path of the object is considered to be part of the name

All backup files are located under the $SYBASE/sybase_dumps directory, which may be on a separate physical disk. These files are kept for a period of 30 day and they are named **database_name_dat.Z** where **database_name** is the database name for that server. For example, a backup of **IoAdAdvService** database would be in a backup file called **IoAdAdvService.dat.Z**.

All SQL script files have the **.sql** suffix. Their names reference the objects they create or functions they perform, and are all located either in $SYBASE/scripts or below. A SQL file to create the **IoAdAdvService** database is named **$SYBASE/scripts/create.databases/IoAdAdvService.sql**, and another file used to alter the database might be called **$SYBASE/scripts/create.databases/alter_IoAdAdvService.sql**.

All SQL Servers are named [machine name]_srvr, backup servers are [machine name]_backup.

All errorlogs are named [machine name]_errorlog, and backup errorlogs are named [machine name]_backup.log.

As one of the most important, yet least applied concepts, naming conventions are presented in this chapter by examples according to the following rules.

Rule 1: Regardless of the length of the name, it should indicate the function and/or content of the object

Rule 2: Only easily understandable abbreviations should be used

Rule 3: Parts of names are separated by underscores "_", only one optional suffix is permitted (appended to the name by a . ".")

Rule 4: The full path of the object is considered to be part of the name

The names of the databases and tables themselves may or may not follow the above rules, these rules are specifically for the DBA to work with SQL Server objects, and files in the UNIX environment.

All **COTS** software is installed in the /usr/ecs/OPS/COTS directory.

All **SYBASE** software is located in the Sybase home directory (**$SYBASE**).

All backups are located in **$SYBASE**/sybase_dumps directory, which may or may not be on a separate physical disk.

Note

It is strongly recommended that backups be stored on a separate physical disk.

The database dumps are kept for a period of 2 days and also stored on a disk by Networker everyday. The database dumps are named as follows:

dbname.dat_YYMMDDHHMM.Z

where MMDDHHMM is the "sortable" eight digit month, day, hour, and minute. For example, on the date this chapter was written, a backup directory called **backups_for_99021100024.Z**

All SQL script files have the extension **.sql** as a suffix. Their names reference the objects they create or functions they perform, and are all located in **$SYBASE**/scripts.

SQL statement must follow precise syntactical and structural rules, and may include only SQL keywords, identifiers (names of databases, tables, or other database objects), operators, and constants. The characters that can be used for each part of a SQL statement vary from installation to installation and are determined in part by definitions in the default character set that version of the server uses.

For example, the characters allowed for the SQL language, such as SQL keywords, special characters, and Transact-SQL extensions, are more limited than the characters allowed for identifiers. The set of characters which may be used for data is much larger and includes all the characters that can be used for the SQL language or for identifiers.

The sections that follow describe the sets of characters that can be used for each part of a statement. The section on identifiers also describes naming conventions for database objects.

SQL Data Characters

The set of SQL data characters is the larger set from which both SQL language characters and identifier characters are taken. Any character in SQL Server's character set, including both single-byte and multibyte characters, may be used for data values.

SQL Language Characters

SQL keywords, Transact-SQL extensions, and special characters such as the comparison operators > and <, can be represented only by 7-bit ASCII values A- Z, a -z, 0-9, and the following ASCII characters:

Identifiers

Conventions for naming database objects apply throughout SQL Server software and documentation. Identifiers can be up to 30 bytes in length, whether or not multibyte characters are used. The first character of an identifier must be declared as an alphabetic character in the character set definition in use on Server.

The @ sign or _ (underscore character) can also be used. The @ sign as the first character of an identifier indicates a local variable.

Temporary table names must either begin with # (the pound sign) if they are created outside tempdb or be preceded by "tempdb..".

Table names for temporary tables that exist outside tempdb should not exceed 13 bytes in length, including the number sign, since SQL Server gives them an internal numeric suffix.

After the first character, identifiers can include characters declared as alphabetic, numeric, or the character $, #, @, _, ¥ (yen), or £ (pound sterling). However, you cannot use two @@ symbols together at the beginning of a named object, as in "@@myobject." This naming convention is reserved for global variables, which are system-defined variables that SQL Server updates on an ongoing basis.

The case sensitivity of SQL Server is set when the server is installed and can be changed by a System Administrator. To see the setting for your server, execute this command: sp_helpsort

Delimited Identifiers

Delimited identifiers are object names enclosed in double quotes. Using delimited identifiers allows you to avoid certain restrictions on object names. You can use double quotes to delimit table, view, and column names; you cannot use them for other database objects.

Delimited identifiers can be reserved words, can begin with non-alphabetic characters, and can include characters that would not otherwise be allowed. They cannot exceed 28 bytes.

Before creating or referencing a delimited identifier, you must execute:

set quoted_identifier on

The names of database objects need not be unique in a database.

However, column names and index names must be unique within a table, and other object names must be unique for each owner within a database. Database names must be unique on SQL Server.

If you try to create a column using a name that is not unique in the table or to create another database object such as a table, a view, or a stored procedure, with a name that you have already used in the same database, SQL Server responds with an error message.

You can uniquely identify a table or column by adding other names that qualify it, that is, the database name, the owner's name, and, for a column, the table name or view name. Each of these qualifiers is separated from the next by a period:

database.owner.table_name.column_name

database.owner.view_name.column_name

The same naming syntax applies to other database objects. You can refer to any object in a similar fashion:

If the quoted_identifier option of the set command is on, you can use double quotes around individual parts of a qualified object name.

Use a separate pair of quotes for each qualifier that requires quotes.

For example, use:

database.owner."table_name"."column_name"

rather than:

database.owner."table_name.column_name"

The full naming syntax is not always allowed in create statements because you cannot create a view, procedure, rule, default, or trigger in a database other than the one you are currently in. The naming conventions are indicated in the syntax as:

[[database.]owner.]object_name or: [owner.]object_name

The default value for owner is the current user, and the default value for database is the current database. When you reference an object in SQL statements, other than create statements, without qualifying it with the database name and owner name, SQL Server first looks at all the objects you own, and then at the objects owned by the Database Owner, whose name in the database is "dbo." As long as SQL Server is given enough information to identify an object, you need not type every element of its name. Intermediate elements can be omitted and their positions indicated by periods:

database..table_name

You must include the starting element, in this case, database, particularly if you are using this syntax when creating tables. If you omit the starting element, you could, for example, create a table named ..mytable. This naming convention prevents you from performing certain actions on such a table, such as cursor updates.

When qualifying a column name and a table name in the same statement, be sure to use the same naming abbreviations for each; they are evaluated as strings and must match or an error is returned.

Identifying Remote Servers

You can execute stored procedures on a remote SQL Server, with the results from the stored procedure printed on the terminal that called the procedure. The syntax for identifying a remote server and the stored procedure is:

[execute] server.[database].[owner].procedure_name

You can omit the execute keyword when the remote procedure call is the first statement in a batch. If other SQL statements precede the remote procedure call, you must use execute or exec. You must give the server name and the stored procedure name. If you omit the database name, SQL Server looks for procedure_name in your default database. If you give the database name, you must also give the procedure owner's name, unless you own the procedure or the procedure is owned by the Database Owner.

If the server name in interfaces is in uppercase letters, you must use it in uppercase letters in the remote procedure call.

In all cases throughout this chapter, when actual examples are provided, those which reference UNIX commands will be preceded by a “%”, and those that reference SQL statements will be preceded by a number and a “>” (e.g. 1>sp_help tablename).

This page intentionally left blank.

ECS Database Environment

The database environment at ECS spans multiple databases serving numerous subsystems across several hosts. Figure 4.1.1-1 shows the system Baseline Hardware/Database Map for the Goddard Space Flight Center (GSFC) Distributed Active Archive Center (DAAC). There are similar mapping diagrams for each of the other DAACs. The diagram, using ECS naming conventions, lists the Subsystem name (i.e., SUB – Subscription Server; SDSRV – Science Data Server; INGEST), the Host Platform (i.e., G0INS01, G0ACG01, G0ICG01), the Sybase Server designation (i.e., g0ins01_srvr, g0acg01_srver, g0icg01_srvr), the Database Names (i.e., SubServer, EcDsScienceDataServer1, Ingest), the various database component sizes (e.g., DB size, Log size, Index size), the Device Type (raw or filesystem) and the Database Owner Names (i.e., css_role, sdsrv_role)

The Hardware/Database Mapping document, 920-TDx-009-Rev*xx*, along with many other key reference documents can be accessed through the Pete.hitc.com website: http://cmdm.east.hitc.com. Select the "ECS Baseline" link button, and then the "Technical Documents" button to view this wealth of system data.

Goddard Space Flight Center (GSFC) DAAC Baseline Hardware / Database Mapping
DROP 4PY and DROP 5A

Subsystem	DAAC Platform	DAAC Sybase Server	DAAC Database Names (1)	DB Size	Log Size	Index Size	Device Type	DBO Names
CLS	N/A	N/A	No databases	N/A	N/A	N/A	N/A	N/A
CSS/DC ACL	N/A	N/A	No databases	N/A	N/A	N/A	N/A	N/A
CSS/DC SUB	G0INS01	g0ins01_srvr	SubServer SubServer_TS1 SubServer_TS2 tempdb master sybsystemprocs	120MB 40MB 40MB 100MB 50MB 60MB	60MB 20MB 20MB N/A N/A N/A	60MB 20MB 20MB N/A N/A N/A	raw raw raw filesystem raw filesystem	css_role css_role css_role
DMS	G0INS02	g0ins02_srvr	EcDmDictService EcDmDictService_TS1 EcDmDictService_TS2 tempdb master sybsystemprocs	360MB 120MB 120MB 100MB 50MB 60MB	180MB 60MB 60MB N/A N/A N/A	180MB 60MB 60MB N/A N/A N/A	raw raw raw filesystem raw filesystem	dms_role dms_role dms_role
DSS/DIST	G0ICG01	g0icg01_srvr	(Moved into STMGT database)	N/A	N/A	N/A	N/A	N/A
DSS/DSFW	N/A	N/A	No databases	N/A	N/A	N/A	N/A	N/A
DSS/SDSRV	G0ACG01	g0acg01_srvr	EcDsScienceDataServer1 EcDsScienceDataServer1_TS1 EcDsScienceDataServer1_TS2 tempdb master sybsystemprocs	4,800MB 1,600MB 1,600MB 1,000MB 50MB 60MB	1,200MB 400MB 400MB N/A N/A N/A	1,200MB 400MB 400MB N/A N/A N/A	raw raw raw filesystem raw filesystem	sdsrv_role sdsrv_role sdsrv_role
DSS/STMGT	G0ACG01	g0acg01_srvr	stmgrdb1 stmgrdb1_TS1 stmgrdb1_TS2	600MB 160MB 160MB	300MB 80MB 80MB	300MB 80MB 80MB	raw raw raw	stmgr_role stmgr_role stmgr_role
INGEST	G0ICG01	g0icg01_srvr	Ingest Ingest_TS1 Ingest_TS2 tempdb master sybsystemprocs	480MB 160MB 160MB 100MB 50MB 60MB	240MB 80MB 80MB N/A N/A N/A	240MB 80MB 80MB N/A N/A N/A	raw raw raw filesystem raw filesystem	ingest_role Ingest_role Ingest_role

Figure 9 . Baseline Hardware/Database Map for the Goddard Space Flight Center (GSFC) Distributed Active Archive Center (DAAC).

Directory Structure

In addition to the fundamental database design, ECS operates on a concept of mutually exclusive, functionally identical modes. The main mode that interacts with live data and customers is called the Operational mode (OPS). Other modes available at the DAACs are nominally called TS2, TS1, and SHARED. The SHARED mode contains files common to all modes. The TS2 and TS1 modes are used to implement and test new functionality for both COTS and CUSTOM code. After modifications are installed and successfully tested in a non-OPS mode, they are promoted to the next mode level and ultimately upgraded into the OPS mode. This concept enables uninterrupted operation for live data and user interaction while simultaneously field-testing new code. Figure 10 shows this multi-mode directory structure. The OPS mode is shown here and is identical for the TS2 and TS1 modes.

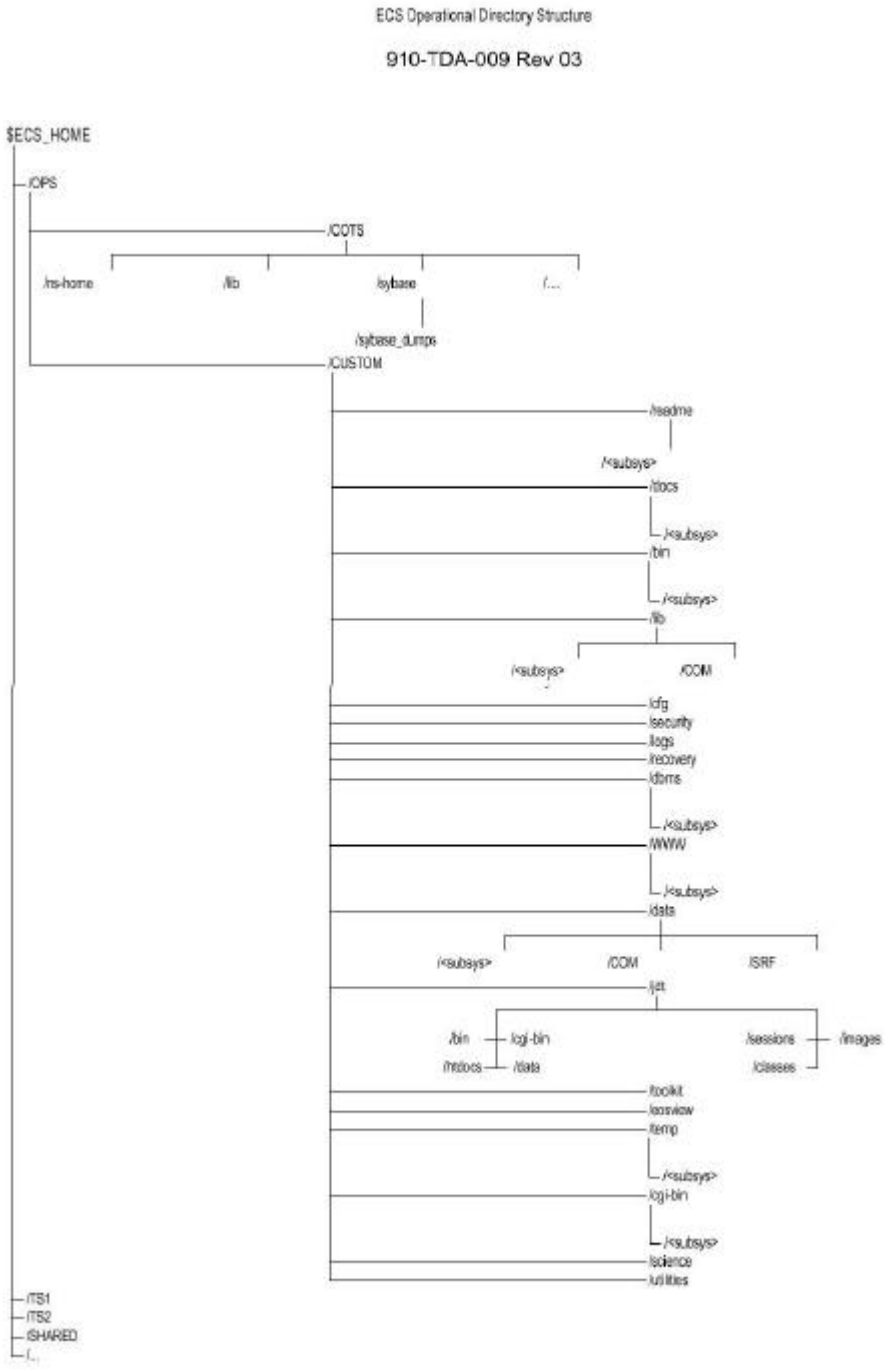


Figure 10. ECS Operational Directory Structure.

The **sybase** directory structure is described in the following table. Subdirectories under the **scripts** can contain template files with easy to modify examples of SQL and SQL command syntax.

The following table describes the directory structure used for Sybase database administration in Release 6.

$SYBASE is a shortcut for the full path to the Sybase home directory. Because each DAAC may support a different full path to that directory (e.g., /usr/ecs/OPS/COTS/sybase, /vol0/sybase, etc.), this document will refer to the path as **$SYBASE**.

Table 6. SYBASE Directory Structure.

Directory	Contains
$SYBASE/bin	Utilities necessary to load, run, and access the server
$SYBASE /install	Files used to start and initialize dataservers, backupserver and to record server messages (errorlogs)
$SYBASE /lib	db-lib, ct-lib, and xa-lib client library files used by applications to gain access to the server (local to server) *__Applications use automounted libraries__.
$SYBASE /scripts	Root directory for all script files executed on the server
$SYBASE /sybase_dumps	Root directory that contains all backup subdirectories, it is recommended, but not required, that this directory be on a separate physical disk. Dumps both database and transaction logs. **Backups are stored on disk in the backup subdirectories.
backup subdirectories $SYBASE /sybase_dumps/dumps $SYBASE /sybase_dumps/trans $SYBASE /sybase_dumps/dumps/logs $SYBASE /sybase_dumps/trans/logs $SYBASE /sybase_dumps/Week1 $SYBASE /sybase_dumps/Week2 $SYBASE /sybase_dumps/Week1/logs $SYBASE /sybase_dumps/Week2/logs	A cron job is run at night to move data from the current (week1) directory to the previous (week2) directory. Then, a dump of the databases and transaction logs is executed and is stored in the current directory. All logs are written to the log directory. Files are saved using the following naming convention:: dbname.dat.YYMMDDHHMM.Z - full database dumps dbname.tran.YYMMDDHHMM.Z - full transaction log dumpsdbname_backup.log. dbname_ERR.log.MMDDHHMM - Error log filesdbname_dbcc.log. MMDDHHMM
**xxdmh02 serves as a remote Backup Server	**xx are the 2 letter codes to identify a DAAC site (i.e., g0 = Goddard)

Sybase Adaptive Server Enterprise

The version of Sybase Adaptive Server Enterprise installed in drop 5B on the HP, SUN and SGI platforms is ASE 11.5.1. Version numbers for this COTS software and all other software products are published in the COTS SOFTWARE VERSION BASELINE REPORT, 910-TDA-003-Rev*xx*. A more complete listing of software and the individual host installation content is provided in the DAAC-specific Hardware/Software Map, document 920-TDx-002-Rev*xx*. As upgrades are released and installed, version status will be reflected in these documents.

Database Schemas

All database designs in ECS are thoroughly documented in the 311 Series of documents, <SUBSYSTEM> Database Design and Schema Specifications for the ECS Project. These individual subsystem documents provide the DBA with a complete description of each database including:

Physical Data Model Entity Relationship Diagram

Tables

Columns

Column Domains

Rules

Defaults

Views

Integrity Constraints

Triggers

Stored Procedures

The Schema documents also provide Performance and Tuning Factors, Database Security information, Scripts, and Entity Relationship Diagram Keys. Figure 11 is a portion of the Entity Relationship Diagram for the INGEST subsystem. To access this document, and the other subsystem 311 Series documents, use the following URL: http://edhs1.gsfc.nasa.gov/. From the ECS Data Handling Homepage select the "Document Catalog" link and then the "Design Documents and Specifications" link. From this point you can select the relevant subsystem document.

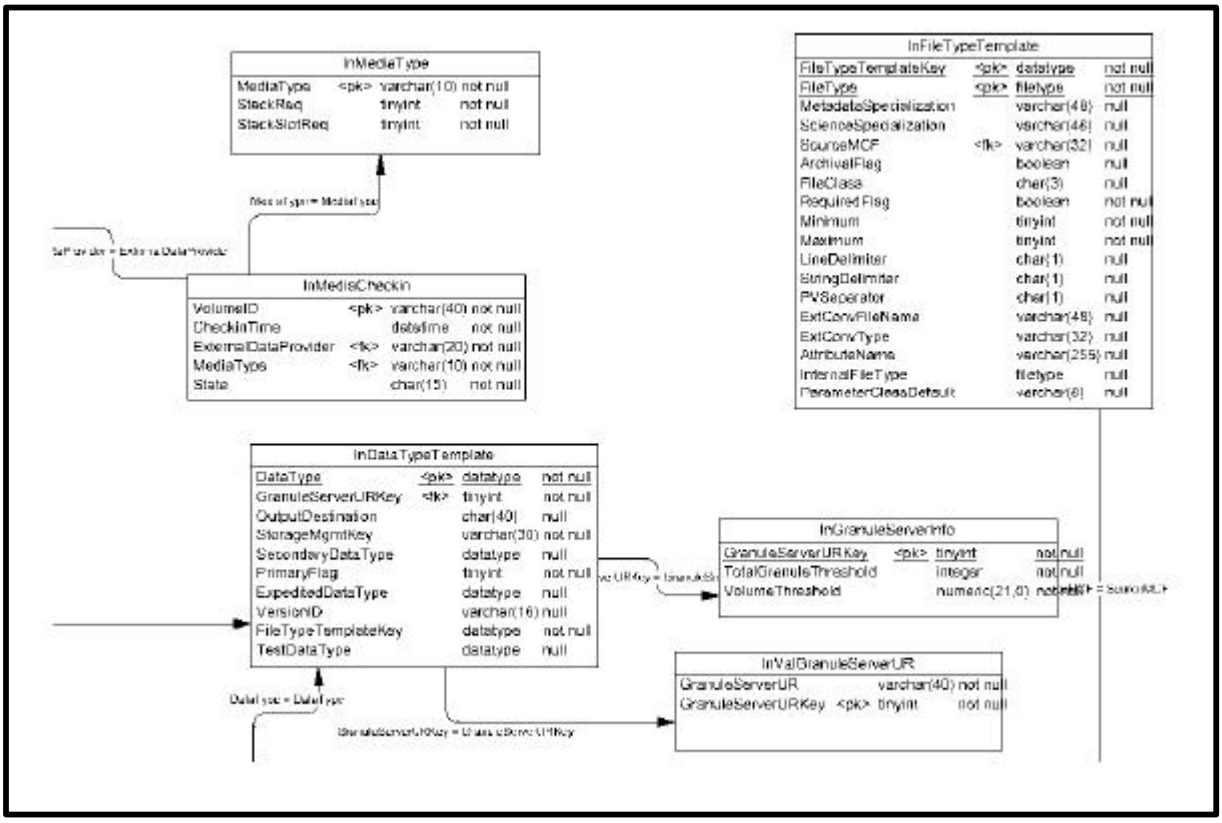


Figure 11. Partial Entity Relationship Diagram - INGEST/

DAAC Database Configurations

The key factors that determine optimum performance for any database are its configuration parameters. A new document available on the Pete Server (http://cmdm.east.hitc.com), SYBASE SQL Server 11.0.x, ALL DAAC Database Configurations (910-TDA-021-Rev00), provides DBAs with detailed data and recommendations for Sybase configurable parameters. In addition to providing default values and the DAAC-specific parameter values for each host, it also describes the Sybase Segment naming conventions to be used at each site for assigning Sybase disk devices to databases. Future versions of this document will capture and baseline the disk devices at each DAAC and the interface file listings at each DAAC.

Figure 12 is an example of DAAC-specific configuration parameters for Goddard Space Flight Center on host g0msh08.

Host Name: g0msh08

Parameter Name	Memory Used	Default Value	Configured Value	Run Value
number of devices	8	10	20	20
number of locks	469	5000	5000	5000
number of remote connections	33	20	20	20
number of remote logins	22	20	20	20
total memory	15000	7500	7500	7500
max online engines	355	1	1	1
number of user connections	2121	25	25	25
procedure cache percent	838	20	20	20
number of open databases	396	12	12	12
number of open objects	489	500	500	500
stack size	2075	34816	34816	34816

Figure 12. DAAC-Specific Configuration Parameters.

Database Disk Partitioning

System documentation also provides graphic depictions of server disk partitioning for all of the hosts (e.g. Ingest, MSS, CSS, PDPS DBMS). The 922-TDx-0xx-Rev*00* series of documents (available on the PETE server) provide a block diagram of the disk partitioning for each of the servers (Figure 12) and also secondary tables (Figure 13) describing the physical break-down of the individual disks including Slice, Start Block, Total Blocks, Start MB, Total MB, XLV Name, Mount, and Type.

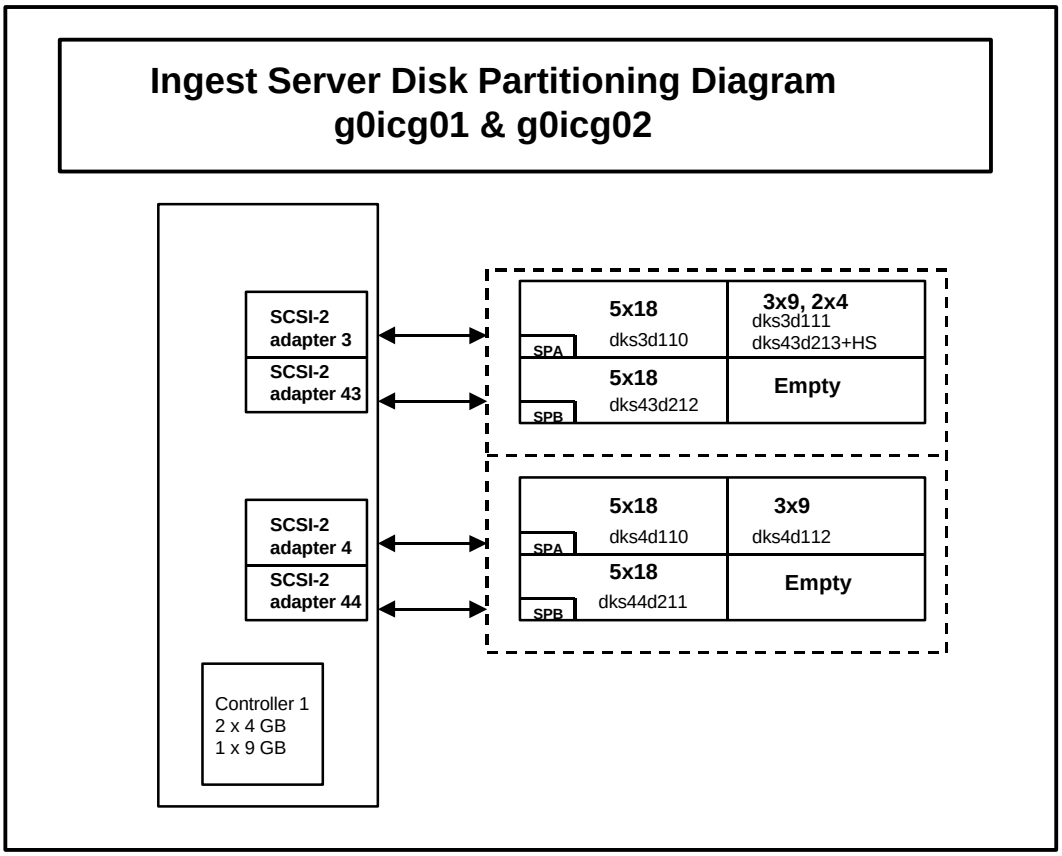


Figure 13. Server Disk Partitioning Block Diagram.

dks3d111, 2x0 GB Raid Level 1, Mirrored

Slice	Start Block	Total Blocks	Start MB	Total MB	XLV Name	Mount	Type
0	2048	1024000	1	1	ingestlog		xlv
1	1026048	512000	501	501	sybsecurity		xlv
2	1538048	256000	751	125	sybmaster		xlv
3	1794048	512000	876	250	sybsecarchive		xlv
4	2306048	512000	1126	250	ddistlog		xlv
5	2814048	1024000	1376	500	stmgtlog		xlv
6	3842048	5120000	1876	2500	sybase_dumps		xlv
7	9862048	8264978	4376	4036	spare1		xlv
8	0	2018	0	1	n/a		Volhdr
10	0	17227026	0	8412	n/a		volume

Figure 14. Ingest Server Disk Partitioning Diagram.

Database Administrator Responsibilities

The DAAC Database Administrators typically perform the following functions:

- Perform the database administration utilities, such as database backup, maintenance of database transaction logs, and database recovery.
- Monitor and tune the physical allocation of database resources.
- Maintain user accounts.
- Create user registration and account access control permissions in the security database.
- Work with data specialists in information management tasks involving databases, data sets, and metadata management.
- Perform daily database synchronization.

The following subsections detail the most common functions.

Start an SQL Server

A SQL server process is **manually** started when a new server is installed or after a system outage.

In order to perform the procedure, the DBA must have obtained the database server name.

To start a SQL server process, the DBA must have sa_role (SQL Server)

Start the SQL Server Process Procedure

1 Log on to a local host.

2 Set display to current terminal by typing: **setenv DISPLAY *IPNumber*:0.0** or **setenv DISPLAY *ServerName*:0.0**, then press **Return**.

3 Log into the the server on which the SQL Server Process is to be started by typing: **/tools/bin/ssh *ServerName***, then press **Return**.

- If you have previously set up a secure shell passphrase and executed **sshremote**, a prompt to **Enter passphrase for RSA key '<*user@localhost*>'** appears; continue with Step 4.
- If you have not previously set up a secure shell passphrase; go to Step 5.

4 If a prompt to **Enter passphrase for RSA key '<*user@localhost*>'** appears, type your ***Passphrase*** and then press the **Enter** key. Go to step 6.

5 At the **<*user@remotehost*>'s password:** prompt, type your ***Password*** and then press the **Enter** key.

6 Log into the server as sybase, type, ***su – sybase*** and press **Return**.

- A password prompt is displayed.

7 Enter ***sybase password***, then press **Return**.

- You are authenticated as yourself and returned to the UNIX prompt.

8 At the UNIX prompt, type, **cd install** and then press **Return**.

9 At the UNIX prompt, type **./RUN_*servername* & *)***, then press **Return**.

10 At the UNIX prompt, type **showserver**, then press **Return**.

- This displays a listing of the SQL Server processes that are running.

Starting the SQL Backup Server Procedure

(NOTE: This step should be done after SQL Server is up and running)

1 Log on to a local host.

2 Set display to current terminal by typing: **setenv DISPLAY *IPNumber*:0.0** or **setenv DISPLAY *ServerName*:0.0**, then press **Return**.

3 Log into the the server on which the SQL Server Process is to be started by typing: **/tools/bin/ssh *ServerName***, then press **Return**.

- If you have previously set up a secure shell passphrase and executed **sshremote**, a prompt to **Enter passphrase for RSA key '<*user@localhost*>'** appears; continue with Step 4.
- If you have not previously set up a secure shell passphrase; go to Step 5.

4 If a prompt to **Enter passphrase for RSA key '<*user@localhost*>'** appears, type your ***Passphrase*** and then press the **Enter** key. Go to step 6.

5 At the **<*user@remotehost*>'s password:** prompt, type your ***Password*** and then press the **Enter** key.

6 Log into the server as sybase, type, ***su – sybase*** and press **Return**.

- A password prompt is displayed.

7 Enter ***sybase password***, then press **Return**.

- You are authenticated as yourself and returned to the UNIX prompt.

8 Type, **cd install** and press RETURN

9 Type, **./RUN_*backupservername* &** and press RETURN

Note: You may have to hit RETURN one more time

Starting the SQL Monitor Server Procedure

(NOTE: This step should be done after the SQL Server is up and running)

1 Log on to a local host.

2 Set display to current terminal by typing: **setenv DISPLAY *IPNumber*:0.0** or **setenv DISPLAY *ServerName*:0.0**, then press **Return**.

3 Log into the the server on which the SQL Server Process is to be started by typing: **/tools/bin/ssh *ServerName***, then press **Return**.

- If you have previously set up a secure shell passphrase and executed **sshremote**, a prompt to **Enter passphrase for RSA key '<*user@localhost*>'** appears; continue with Step 4.
- If you have not previously set up a secure shell passphrase; go to Step 5.

4 If a prompt to **Enter passphrase for RSA key '<*user@localhost*>'** appears, type your ***Passphrase*** and then press the **Enter** key. Go to step 6.

5 At the **<*user@remotehost*>'s password:** prompt, type your ***Password*** and then press the **Enter** key.

6 Log into the server as sybase, type, ***su – sybase*** and press **Return**.

- A password prompt is displayed.

7 Enter ***sybase password***, then press **Return**.

- You are authenticated as yourself and returned to the UNIX prompt.

8 Type, **cd SMS11.x/bin** and press RETURN

9 Type, **./monserver –S*server-name* –M*monitor-server-name* –Usa &** and press RETURN

Stop an SQL Server

A SQL server process is stopped by the DBA when the system to be brought down for maintenance. The SQL Backup and Monitor servers should be stopped before stopping the SQL server. The **shutdown** command issued from within ISQL shuts down the SQL server on which you are logged in. It allows for the completion of any current SQL processes and blocks the start of any new SQL processes before the server is shutdown. Adding a **nowait** option to the command immediately terminates all processes and shuts down SQL server. Using the **nowait** option may result in loss of data and a more complicated recovery procedure, so use it sparingly.

Stopping the SQL Backup Server Procedure

(NOTE: This step should be done before shutting down the SQL Server)

---

1 Log on to a local host.

2 Set display to current terminal by typing: **setenv DISPLAY *IPNumber*:0.0** or **setenv DISPLAY *ServerName*:0.0**, then press **Return**.

3 Log into the the server on which the SQL Server Process is to be stopped by typing: **/tools/bin/ssh *ServerName***, then press **Return**.

- If you have previously set up a secure shell passphrase and executed **sshremote**, a prompt to **Enter passphrase for RSA key '<*user@localhost*>'** appears; continue with Step 4.
- If you have not previously set up a secure shell passphrase; go to Step 5.

4 If a prompt to **Enter passphrase for RSA key '<*user@localhost*>'** appears, type your ***Passphrase*** and then press the **Enter** key. Go to step 6.

5 At the **<*user@remotehost*>'s password:** prompt, type your ***Password*** and then press the **Enter** key.

6 Log into the server as sybase, type, ***su – sybase*** and press **Return**.

- A password prompt is displayed.

7 Enter ***sybase password***, then press **Return**.

- You are authenticated as yourself and returned to the UNIX prompt.

8 Type, **setenv SYBASE *sybase-directory-path***

9 Type, **setenv DSQUERY *sql-servername***

10 Type, **set path = ($path $SYBASE/bin $SYBASE/install $SYBASE)**

11 **Note:** GDAAC implementation has a sybsetup.csh file (type source sybsetup.csh)

12 Type, **isql –U<username>** and press RETURN

13 Type, the ***password*** when prompted for it and press RETURN

14 Type, **shutdown SYB_BACKUP** at "**1>**" prompt and press RETURN

15 Type, **go** at "**2>**" prompt and press RETURN

Stopping the SQL Monitor Server Procedure

(NOTE: This step should be done before the SQL Server is shutdown)

---

1 Log on to a local host.

2 Set display to current terminal by typing: **setenv DISPLAY *IPNumber*:0.0** or **setenv DISPLAY *ServerName*:0.0**, then press **Return**.

3 Log into the the server on which the SQL Server Process is to be stopped by typing: **/tools/bin/ssh *ServerName***, then press **Return**.

- If you have previously set up a secure shell passphrase and executed **sshremote**, a prompt to **Enter passphrase for RSA key '<*user@localhost*>'** appears; continue with Step 4.
- If you have not previously set up a secure shell passphrase; go to Step 5.

4 If a prompt to **Enter passphrase for RSA key '<*user@localhost*>'** appears, type your ***Passphrase*** and then press the **Enter** key. Go to step 6.

5 At the **<*user@remotehost*>'s password:** prompt, type your ***Password*** and then press the **Enter** key.

6 Log into the server as sybase, type, ***su – sybase*** and press **Return**.

- A password prompt is displayed.

7 Enter ***sybase password***, then press **Return**.

- You are authenticated as yourself and returned to the UNIX prompt.

8 Type, **setenv SYBASE *sybase-directory-path***

9 Type, **setenv DSQUERY *sql-servername***

10 Type, **$SYBASE/bin/isql –Usa** and press RETURN

11 Type, the ***sa-password*** when prompted for it and press RETURN

12 Type, **sms_shutdown** at "**1>**" prompt and press RETURN

13 Type, **go** at "**2>**" prompt and press RETURN

Stopping the SQL Server Procedure

1 Log on to a local host.

2 Set display to current terminal by typing: **setenv DISPLAY *IPNumber*:0.0** or **setenv DISPLAY *ServerName*:0.0**, then press **Return**.

3 Log into the the server on which the SQL Server Process is to be stopped by typing: **/tools/bin/ssh *ServerName***, then press **Return**.

- If you have previously set up a secure shell passphrase and executed **sshremote**, a prompt to **Enter passphrase for RSA key '<*user@localhost*>'** appears; continue with Step 4.
- If you have not previously set up a secure shell passphrase; go to Step 5.

4 If a prompt to **Enter passphrase for RSA key '<*user@localhost*>'** appears, type your ***Passphrase*** and then press the **Enter** key. Go to step 6.

5 At the **<*user@remotehost*>'s password:** prompt, type your ***Password*** and then press the **Enter** key.

6 Log into the server as sybase, type, ***su – sybase*** and press **Return**.

- A password prompt is displayed.

7 Enter ***sybase password***, then press **Return**.

- You are authenticated as yourself and returned to the UNIX prompt.
- You are authenticated as yourself and returned to the UNIX prompt.
- Your new home directory is **/usr/ecs/OPS/COTS/sybase** and all required environment variables have been set.

8 Type, **setenv SYBASE *sybase-directory-path***

9 Type, **setenv DSQUERY *sql-servername***

10 Type, **set path = ($path $SYBASE/bin $SYBASE/install $SYBASE)**

11 **Note:** GDAAC implementation has a sybsetup.csh file (type source sybsetup.csh)

12 Type, **isql –U<username>** and press **Return**.

13 Type, the **<*password*>** when prompted for it and press **Return**.

14 At the ISQL prompt, type, **shutdown with nowait** at "**1>**" prompt and press **Return**; type **go** at "**2>**", then press **Return**.

15 At the ISQL prompt, type **showserver**, then press **Return**; type **go**, then press **Return**.

- This displays a listing of the SQL Server processes that are running.
- There should be no SQL Server processes running.

17 At the ISQL prompt, type **exit**, then press **Return**.

- You are returned to a UNIX prompt.

Shutdown of SQL Server

Use **shutdown** to bring the server to a halt. This command can only be issued by the Sybase System Administrator (sa).

Syntax:1> **shutdown [backup_server_name]]** [with] [wait] [with nowait]

2> **go**

The "with wait" is the default option. This option brings SQL Server down gracefully.

The "with nowait" option shuts down the SQL Server immediately without waiting for currently executing statements to finish.

If you do not give a server name, shutdown shuts down the SQL Server you are using.

When you issue a shutdown command, SQL Server:

Disables logins, except for System Administrators

Performs a checkpoint in each database, flushing pages that have changed from memory to disk

Waits for currently executing SQL statements or procedures to finish

In this way shutdown minimizes the amount of work that automatic recovery must do when you restart SQL Server.

To see the names of the Backup Servers that are accessible from your SQL Server, execute

sp_helpserver. Use the value in the name column in the shutdown command. You can only shut down a Backup Server that is:

Listed in sysservers on your SQL Server, and

Listed in your local interfaces file.

Note 1

It recommended that "with wait" option be used. This allows executing statements to finish.

Also it is recommended that you perform a checkpoint of all database prior to shutdown.

Note 2

SQS server should be started after the SQL Server

Allocation of Resources

SQL Server can make reasonable default decisions about many aspects of storage management, such as where databases, tables, and indexes are placed and how much space is allocated for each one. However, the System Administrator has ultimate control over the allocation of disk resources to SQL Server and the physical placement of databases, tables, and indexes on those resources.

Loading a database you have created into a different database

Occasionally, you may want to create an exact copy of a database of you system. First, dump the existing database. Then create a database to load with this dump. The database does not have to be the same size as the original. The only requirement is that the destination database must be at least as large as the dumped database and have the same beginning fragments as the original database. This information can be obtained from saved database creation scripts, or by running the following command:

select segmap,'Size in MB'=size/512 from sysusages where dbid= db_id("database_name")

Example:

suppose your database was created with the following statement:

create database dbname on datadevice1 = 1000,

log on Logdevice1 = 200

go

alter device dbname on datadevice2 = 500 running:

select segmap,'Size in MB'=size/512 from sysusages

where dbid= db_id("dbname")

would return:segmap Size in MB

3 1000

4 200

3 500

You could create a 3GB database as follows and load your database into it (using "for load" option will shorten database load time):

create database newdatabase on datadevice3 = 1000 log on logdevice3 = 200

for load

go

alter database newdatabase on datadevice 3=500 for load go

alter database newdatabase on datadevice4=300 for load go

alter database newdatabase on datadevice5=1000 for load go

load database newdatabase from dbname_dump go

Monitoring Space Usage

Thresholds

Thresholds are defined on segments to provide a free space value at which a procedure is executed to provide a warning or to take remedial action.

Use **sp_addthreshold** to define your own thresholds:

sp_addthreshold database_name, segment_name, free_space, procedure_name

where free_space is the number of free pages at which the threshold procedure executes; procedure_name is the stored procedure which the threshold manager executes when the number of free pages falls below the free_space value. Please see the section on Auditing later in this chapter for an example of Thresholds.

Example of Threshold Commands mentioned above:

Sp_addthreshold CustomerDB, "default", 10230, CustDefaultSegWarn

Creating and Managing Logins and Roles

Earlier versions of SQL Server administrative responsibilities needed to be executed by and individual logged in –literally- as sa. Now specific user logins can be assigned components of administrative responsibility, enabling you to track and audit administrative activities.

The three roles are sa_role (systems administrator) for administrative tasks, sso_role(site security officer) for security tasks, and oper_rol (operator) for backup and recovery tasks.

In order to connect to a SQL Server a login must be created by the System Administrator or a system security officer. Login details are stored in the syslogins table in the **master** database.

The system stored procedure **sp_addlogin** adds new login names to the server but does not grant access to any user database.

Syntax: **sp_addlogin** login_name, password, [,default database ,language, fullname]

In order to gain access to a database, the System Administrator, system security officer, of the specific database owner must "add" the user with the **sp_adduser** system stored procedure.

Syntax: 1> **sp_adduser**login_name [username, group_name]
2> go

Example of Creating a Login and Granting Database Access

The DBA has received a request to authorize John Q. Public to the pdps_db_ops database.

***It is a good practice to have a default_db, when you create a user account.**

1. In the **$SYBASE**/scripts/create.users directory, DBA creates a script file containing the sp_addlogin command (named public.sql)

Syntax: % cd /usr/ecs/OPS/COTS/sybase/scripts/create.users

% cp template.sql public.sql

2. DBA modifies appropriate fields so that the script resembles the following:

1> sp_addlogin jpublic,jpublic, default_db

2> go

3> use pdps (OPS mode) 4> go

5> sp_adduser jpublic

6> go

7> sp_helpuser

8> go

3. DBA runs the script from the UNIX command prompt:

Syntax: % isql -Usa -S**servername** - public.sql -opublic.out

4. DBA checks the public.out file for success

Permissions

Permissions are used to control access within a database. The DBA uses the **grant** and **revoke** statements to accomplish this. There are two types of permissions within a database, **Object** and **Command**. In general, **Object** privileges control select, insert, update, delete, and execute permissions on tables, views, and stored procedures. **Command** permissions control access to the **create** statements for databases, defaults, procedures, rules, tables, and views.

The syntax for the **grant** and **revoke** statements are quite similar:

grant {all [privileges] | command_list }

to { public | name_list | role_name }

revoke {all [privileges] | command_list }

from { public | name_list | role_name }

Example of Granting Privileges to a Specific User

The DBA receives a request that John Q. Public should be able to read the DATA_INFO table and read and update the SUBSCRIPTION_NOTIFICATION TABLE.

Syntax: 1> **grant** select on DATA_INFO to jpublic

2> **grant** select, update on SUBSCRIPTION_NOTIFICATION to jpublic

go

Note: It is recommended that the DBA store these command in a ".sql" file in the **$SYBASE**/scripts/create.db_objects directory, along with their results.

Backup and Recovery

Database and transaction log backups and recoveries are probably the most important tasks the Database Administrator must perform. Without regular and complete backups of databases and transaction logs, the potential for enormous amounts of data to be lost is very great.

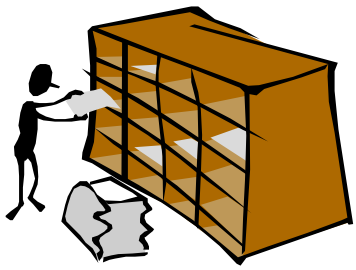
Frequency and Schedule

Each DAAC is responsible for determining its own backup schedule to meet its individual requirements. Nonetheless, it is strongly suggested that a regular schedule of database backups be established and maintained. Although the databases are included in the daily backups of the entire system, separate database backups should be performed nightly.

Database backups are run by a UNIX **cron** job which executes the **sybasedump** program located in the **$SYBASE** directory. No intervention in the automatic backup process is required by the DBA, though periodic checks of the Backup subdirectories are recommended.

Manual backups can be performed at any time by the DBA and are recommended for the following situations:

- Any change to the **master** database, including new logins, devices, and databases.
- Any major change to user databases, such as a large ingest or deletion of data, definition of indexes, etc.,
- Other mission-critical activities as defined by the DAAC operations controller.



Manual Database Backup Procedure

1 Log on to a local host.

2 Set display to current terminal by typing: **setenv DISPLAY *IPNumber*:0.0** or **setenv DISPLAY *ServerName*:0.0**, then press **Return**.

3 Log into the server which contains the database to be backed up by typing: **/tools/bin/ssh *gsfcsparc5.gsfcmo.ecs.nasa.gov***, then press **Return**.

- If you have previously set up a secure shell passphrase and executed **sshremote**, a prompt to **Enter passphrase for RSA key '<*user@localhost*>'** appears; continue with Step 4.
- If you have not previously set up a secure shell passphrase; go to Step 5.

4 If a prompt to **Enter passphrase for RSA key '<*user@localhost*>'** appears, type your ***Passphrase*** and then press the **Enter** key. Go to step 6.

5 At the **<*user@remotehost*>'s password:** prompt, type your ***Password*** and then press the **Enter** key.

6 Log into Sybase by typing: **su-sybase**, then press **Return.**

- A password prompt is displayed.

7 Enter ***SybasePassword***, then press **Return**.

- Remember that ***SybasePassword*** is case sensitive.
- You are authenticated as yourself and returned to the UNIX prompt.
- Your new home directory is **/usr/ecs/OPS/COTS/sybase** and all required environment variables have been set.

8 At the UNIX prompt, type **isql -Usa**, then press **Return**.

9 To backup the database, at the ISQL prompt, type **dump database *DBName* to *BackupDirectory/DBName*.dat**, then press **Return**; type **go**, then press **Return**.

10 To backup the transaction log, at the ISQL prompt, type **dump transaction *DBName* to *BackupDirectory/DBName*_tx.dat**, then press **Return**; type **go**, then press **Return**.

Database Recovery

A database recovery is performed when the Database Administrator determines that some database data is corrupt, or when the System Administrator determines that a database device has failed. The System Administrator makes a request to the Database Administrator, who performs the restore and notifies the System Administrator when the restore is complete.

The symptoms of media failure are as variable as the causes. If only a single block on the disk is bad, your database may appear to function perfectly for some time after the corruption appears; this is why you should run **dbcc** commands frequently. If an entire disk or disk controller is bad, you will not be able to use a database. SQL Server marks it as suspect and displays a warning message. If the disk storing the master database fails, users will not be able to log into the server, and users already logged in will not be able to perform any actions that access the system tables in master.

The complete database device restoration procedure is actually a suite of procedures that must be performed in the prescribed order:

1. The device failure has been verified by the System Administrator and requests restoration from the DBA.
2. If possible, perform a dump of the current database and the current database transaction log for each database on the failed device to be restored. If this is not possible due to the damage to the database device, then the DBA will have to use the most recent database and transaction log dumps.
3. If possible, the DBA examines the space usage for each database on the failed device. The same defaults should be set when the new database device(s) is(are) created.
4. The DBA drops the database(s) on the failed device, then the database device itself is dropped.
5. The DBA initializes a new database device. This is why it is important to keep the device creation scripts.
6. The DBA recreates each user database on the new database device. This is why it is important to keep the user creation scripts.
7. Each database is reloaded with data from the database backups and the transaction log backups. It is this procedure that is detailed below.
8. The DBA notifies the System Administrator when the database restoration is complete.

In order to perform the procedure, the DBA must have obtained the following information:

- Name of database device which has failed
- Name of replacement device
- Name of the databases which reside on the failed device
- Name of the backup volumes
- Name of the dump files on the backup volumes

Table 7. Backup and Recovery Definitions.

Term	Definition
Backup Script Components	Located in the **$SYBASE** directory, they include: sybasedump, dmpdb_trns, copy_daily_dumps_to_week1, copy_daily_dumps_to_week2
Backup files	Defined in Table 4.2-2, the location of these files has been determined during server setup
Backup Statements	Generated from the sql in sybasedump these include calls to dbcc, Dump Database, and Dump Transaction commands
Backup Subdirectory	The only directory level underneath of the Backup Directory, defined in Table 4.2-2.
Backup Summary	An extraction of the successful Dump messages along with any errors generated by the Backup Statements stored in the Backup Subdirectory.

Automatic Backups

The following are the list of all procedures and scripts files that are currently being used for Sybase backups. There are cron jobs running at all sybase **servers** that have SQL server installed. All dump files are currently written to LOCAL machine. The site DBA is responsible for configuring the backup dump to the REMOTE sybase directory.

To check if the crontab is up and running, enter:

> crontab -l

Example of the output:

019 * * 1-6 /usr/ecs/OPS/CUSTOM/dbms/COM/DBAdmin/EcCoDbSyb_DumpDb

012 * * 1-6 /usr/ecs/OPS/CUSTOM/dbms/COM/DBAdmin/EcCoDbSyb_DumpTran

021 * * 1-5 /usr/ecs/OPS/CUSTOM/dbms/COM/DBAdmin/EcCoDbSyb_CkErrorLog

NOTE:

If the crontab is not running enter:

> crontab /usr/ecs/OPS/COTS/sybase/run_sybcron

The following files will be installed by EcCoAssist to the /usr/ecs/OPS/CUSTOM/dbms/COM/DBAdmin directory:

EcCoDbSyb_README

EcCoDbSyb_DumpDb

EcCoDbSyb_DrumpTran

EcCoDbSyb_DbStat

EcCoDbSyb_SedFile

EcCoDbSyb_DboMail

EcCoDbSyb_SetupKsh

EcCoDbSyb_CkErrorLog

EcCoDbSyb_tran_log.awk

SCRIPTS	**DESCRIPTIONS**
EcCoDbSyb_SetupKsh	This file contains the SYBASE and DSQUERY (server) environment setup. This file is call by EcCoDbSyb_DumpDb, EcCoDbSyb_DrumpTran, and EcCoDbSyb_CkErrorLog scripts.
EcCoDbSyb_DumpDb	This script contains the code to dump the databases. First, it checks for any DBCC error on the master database, if there is any error on the master, the script sends an email to the DBA and exit

	the program. If the master database dump was successfull, then the rest of the databases are dumped. Each database has a DBCC check, if there is any error on the database then the database is NOT dumped and an email is send to the DBA. At the end, an status email is send, providing all the database names that were succefully dumped
EcCoDbSyb_DumpTran	This script contains the code to dump the transaction logs. This dumps the transaction logs for each database, it check the error log file, if the error Msg is 4207 or 4221 it will do a dump of the database firt, then it will do the trasaction dump. If there is any other error Msg then the transaction dump will fail and email will be send. At the end, an status of the transaction log dumps is email to the DBA
EcCoDbSyb_SedFile	This file contains all the database that don't need to be dump (i.e., temp, model, etc.)
EcCoDbSyb_DboMail	This file contains the email list of all the DBA's.
EcCoDbSyb_DbStat	This script updates the index table of a database. This script is called from EcCoDbSyb_DumpDb after each successfully database dump.

(Continued)

EcCoDbSyb_CkErrorLog	This script checks for specific database error messages from the Sybase Error Log File every hour and emails the error messages to the DBA's in the EcCoDbSyb_DboMailfile.
EcCoDbSyb_tran_log.awk	This script matches the current hour with the hour the error messages were enerated in the Error Log File. If errors found, the messages are saved in a mailfile and sent to DBA's.

THE FOLLOWING FILES MUST BE MODIFIED BEFORE RUNNING ANY OF THE ABOVE SCRIPTS:

EcCoDbSyb_SetupKsh	Make user you have the SYBASE files under /usr/ecs/OPS/COTS/sybase
EcCoDbSyb_SedFile	Add any other database that might not need to be backed up. The databases that are listed in this file do not need to be backed up.
EcCoDbSyb_DboMail	Add/delete the email of the DBA and any other email that might need to be added/deleted. All the errors and status will be send to them.
run_sybcron	The following is an example on the crontab file that should be run by a sybase user. The first one will run the EcCoDbSyb_DumpDb script that dumps the databases at midnight from Monday to Saturday. The second one, EcCoDbSyb_DumpTran script that dumps the transaction logs will run tree times a day, 10AM, 1PM and 4PM from Monday to Saturday. The Third one, EcCoDbSyb_CkErrorLog that check the SYBASE error log file will run every hour from Monday to Saturday.

0 0 * * 1-6 /usr/ecs/OPS/CUSTOM/dbms/COM/DBAdmin/EcCoDbSyb_DumpDb

0 10,13,16 * * 1-6 /usr/ecs/OPS/CUSTOM/dbms/COM/DBAdmin/EcCoDbSyb_DumpTran

0 * ** 1-6 /usr/ecs/OPS/CUSTOM/dbms/COM/DBAdmin/EcCoDbSyb_CkErrorLog

NOTE: Make sure there is an OPS mode directory with all script files.

All these scripts reside in "/usr/ecs/OPS/CUSTOM/dbms/COM/DBAdmin" directory. The assigned site DBA will be responsible for maintaining, modifying and applying necessary changes that are applicable to their site as for (security, and backup schedule).

SQL Server backups are performed nightly by a **cron** job which runs the **run_sybcron** program located in the **$SYBASE**/ directory. The following table of definitions will be used throughout the rest of this section.

Table 8. Automatic Backup Components.

Component Name	Function(s)
run_sybcron	File added with the crontab -e command, contains several executable cron commands. **Example:** 00 19 * * 1-6 /data1/COTS/sybase/sybasedump
EcCoDbSyb_DumpDb	Controlling script that performs the following functions: run isql to create the Backup Statements run isql to execute the Backup Statements record the results of the Backup Statements in Backup Files copy the Backup Files to the Backup Subdirectory create the Backup Summary "greps" successful Dump statements along with any errors generated, sends e-mail to the DBA and writes them to the backup_summary file
sp	SQL Server password file - contains password for backup role

No intervention in the Automatic Backup Process is required by the DBA, though periodic checks of the Backup Subdirectories are recommended.

Manual Backups

Manual backups can be performed at any time by the System Administrator and are recommended for the following situations:

Any change to the **master** database - this includes new logins, devices, and databases

Any major change to user databases - a large ingest or deletion of data, definition of indexes

Other mission-critical activities - as defined by the DAAC Operations Supervisor.

Both the **dump database** and **dump transaction** command processing are off-loaded to the backup server, and will not affect normal operations of the database. These commands are performed by the System Administrator on appropriate databases as follows:

Syntax:

1> dump database master to
"/usr/ecs/OPS/COTS/sybase/sybase_dumps/dumps/dbname.dat.MMDDHHMM."

go

After dumping the database, compress the dump file by executing:

%compress
/usr/ecs/OPS/COTS/sybase/sybase_dumps/dumps/dbname.dat.MMDDHHMM.

Syntax:

dump transaction pdps_db_ops to
“/usr/ecs/OPS/COTS/sybase/sybase_dumps/trans/pdps_OPS.tran.YYMMDDHHMM""

go

Manual Recovery

Manual recovery of a user database is performed by the System Administrator by the use of the **load database** and **load transaction** commands. For issues concerning the **master** database, please consult your System Administrator’s Guide for assistance. It is recommended that any user database to be recovered be dropped and created with the **for load** option., The **databasename**.sql along with any **alter.databasename**.sql scripts can be , combined into one script which will re-create the user database with the **for load** option. This will insure the success of the **load database** and **load transaction** commands.

The BulkCopy Utility

The **bcp** utility is located in the **$SYBASE**/bin directory and is designed to copy data to and from SQL Server databases to operating system files.

Requirements for Using bcp

In general, you must supply the following information for transferring data to and from SQL Server:

Name of the database and table

Name of the operating system file

Direction of the transfer (in or out)

In order to use **bcp**, you must have a SQL Server account and the appropriate permissions on the database tables and operating system files that you will use. To copy data **in**to a table, you must have **insert** permission on that table. To copy data **out** to an operating system file, you must have select permission on the following tables:

The table being copied

sysobjects

syscolumns

sysindexes

bcp Syntax

bcp [[database_name].owner.]table_name {in | out} datafile [-e errfile] [-n] [-c]

[-t field_terminator] [-r row_terminator] [-U username] [-S server]

Bulkcopy example procedure

1 Log on to a local host.

2 Set display to current terminal by typing: **setenv DISPLAY *IPNumber*:0.0** or **setenv DISPLAY *ServerName*:0.0**, then press **Return**.

3 Log into the server which contains the database to be backed up by typing: **/tools/bin/ssh *ServerName***, then press **Return**.

- If you have previously set up a secure shell passphrase and executed **sshremote**, a prompt to **Enter passphrase for RSA key '<*user@localhost*>'** appears; continue with Step 4.
- If you have not previously set up a secure shell passphrase; go to Step 5.

4 If a prompt to **Enter passphrase for RSA key '<*user@localhost*>'** appears, type your ***Passphrase*** and then press the **Enter** key. Go to step 6.

5 At the **<*user@remotehost*>'s password:** prompt, type your ***Password*** and then press the **Enter** key.

6 Set Login as sybase and cd to "/usr/ecs/OPS/COTS/sybase/scripts/server.bcp

7

- Remember that ***SybasePassword*** is case sensitive.
- You are authenticated as yourself and returned to the UNIX prompt.
- Your new home directory is **/usr/ecs/OPS/COTS/sybase** and all required environment variables have been set.

8 Copy the bcp_script template to [table name]_out.

9 Modify the script with the correct database name, table name, direction, and login name.

10 Run the [table name]_out script and press RETURN for each of the prompts except the last.

11 At the last prompt, store [your responses] in a [table name].fmt file. This creates a format file for future bulkcopy activity.

12 To copy the data to another already created table, repeat steps 6,7, and 8 with the following changes:

- the direction is "in"
- use the optional -f [format file name]

Example of User Database Recovery

The database **UserDB** was created using the following script excerpt: (stored in home/scripts/create.databases/userdb.sql)

create database UserDB on data_dev1 = 100 log on tx_log1 = 50 [with override]

and was modified using the following script excerpt:
(home/scripts/create.databases/alteruserdb.sql)

Alter database **UserDB** on data_dev1=50

For the purposes of this example, the full database backup and transaction log dumps were successful and located in /usr/ecs/OPS/COTS/UserDB.dat and UserDB_tx.dat

1. In the **$SYBASE**/scripts/create.databases directory, DBA makes a script file from the template.

Syntax: % cd /usr/ecs/OPS/COTS/sybase/scripts/create.databases

% cp template.sql userdb_for_load.sql

2. Appropriate items are modified so that the script file resembles the following:

1> create database UserDB on data_dev2=100 log on tx_log2=50 **for load**

2> go

3> alter database UserDB on data_dev3=50

4> go

3. DBA saves the script in $SYBASE/scripts/create.databases/userdb_for_load.sql

4. DBA runs the script from the UNIX command prompt.

Syntax: %isql -Usa -S**servername** -iuserdb_for_load.sql -ouserdb_for_load.out

5. DBA checks the userdb_for_load.out file for success

6. DBA loads the database from the full backup.

Syntax: 1> load database UserDB from

"/usr/ecs/OPS/COTS/sybase/sybase_dumps/week1/dbname.dat.MMDDHHMM"

go

7. DBA loads the transaction file from the transaction file dump.

Syntax: 1> load transaction UserDB from

"/usr/ecs/OPS/COTS/sybase/sybase_dumps/week1/dbname.tran.MMDDHHMM"

3> go

Database Performance and Tuning

Once your application is up and running, the DBA monitors its performance, and may want to customize and fine-tune it. Use the following software tools provided by SQL Server:

Setting query processing options with the **set** command

Setting database options with **sp_dboption**

Monitoring SQL Server activity with **sp_monitor**

Using **update statistics** to ensure that SQL Server makes the best use of existing indexes

Changing system variables using **sp_configure** and the **reconfigure** command

Placing objects on segments to spread i/o, improve throughput, etc. as described in section 4.4.4

For a complete discussion of issues related to SQL Server performance and tuning, refer to your SYBASE SQL Server Performance and Tuning.

Installation of the Applications

DBA should have physical devices configured before installing either autosys or remedy. Both applications use Sybase as their database.

Installation of the Application Database

The installation of the application databases has been automated using ECS Assistant. The application databases are created using the DbBuild script which can only be invoked through ECS Assistant or the Command Line. Scripts that ECS Assistant invokes are:

DbBuild - Create new empty database and loads with initial data

DbPatch - Upgrade to new schema while retaining existing data.

The AUTOSYS Application and other Configuration Issues

The AUTOSYS application works in tandem with PDPS/DPSs to schedule the jobs that run on Science Processor. Autosys installation is performed in /usr/ecs/OPS/COTS by the auto install program located in the autosys/install directory. The results of the installation are stored in an autosys_install.scr file located in the AUTOSYS home directory (/use/ecs/OPS/COTS/autosys). For pdps to run properly with AUTOSYS, the following activities are completed:

A user is defined named **autosys**

autosys user is added to the pdps database (OPS mode)

The autosys server is added to the sysservers table with **sp_addserver**

The server is added to the sysservers table on the AUTOSYS server with **sp_addserver**

Spatial Query Server (SQS)

SQS is a multi-threaded, Sybase Open Query database engine, which is required by the Science Data Subsystem (SDSRV). This product allows definition of spatial data types, spatial operators, and spatial indexing. SQS communicates with Sybase SQL Server to process SDSRV requests to push and pull metadata. SDSRV database server resides on an SGI machine. SQS also, reside on the same machine as SDSRV Sybase SQL Server.

Named X1acg01 - where X is the DAAC specific identifying character.

pathname - /usr/ecs/OPS/COTS/sqs222/bin/sqsserver

Should have one dedicated CPU per instance running. Defaults to one instance now, but may require additional instances later for performance reasons.

Requires one entry in the Sybase "interfaces" file per instance of the SQS server to be run.

Consult startup scripts in /etc/init.d/sybase and /etc/init.d/sqs_222

SQS requires a Sybase login with SA or sa_role and associated password to start. SQS environment variables requirements:

SYBASE = Location of the Sybase home directory. Example: /tools/sybOCv(TBD)

PATH = Must include in this order - /usr/bin; /usr/sbin;$SYBASE/bin

DSQUERY = Name of SQL Server to which to connect. From the $SYBASE/intefaces file. Examples - g0acg01 _srvr

DSLISTEN = Name of SQS server to use. Example - g0acg01_ srvr

SQSUSER = Name of the user (SA or sa_role) for system connection.

SQSPASSWORD = Password for the system connection login

The SQS startup script requires the following information:

SQSHOME = location of sqsserver binaries.

The following is a list of options that can be imbedded in the startup script, these options are beneficial, but they are not required.

SQS STARTUP OPTIONS:

-e path of the SQS server logfile. Example /usr/ecs/OPS/COTS/sqs222/sqs/bin/sqs_222.log

-u number of concurrent SQS connections. Recommend minimum of 125. Example -u 125

Usually started with a delay, after the SQL Server is started. This delay be sufficient for the SQL server to recover and come-up.

$SQSHOME/bin/sqsserver -e $SQSHOME/sqs_222.log -u $USER &

SQS has dependencies on Sybase, such as:

Sybase must be running prior to starting SQS

SQS user id that starts SQS, which is different from the application user ID must have admin privileges

SQS opens a connection to Sybase's because it writes to the Sybase System tables

SQS server thread runs under the userid sa. In order to avoid confusion when monitoring this thread, it is best to:

create a separate login and userid specifically to monitor SQS

grant sa_role authority to the userid created to monitor SQS

EXAMPLE: 1> **sp_adduser** sqs_mon

grant sa_role to sqs_mon

go

Configuration of XLV partitions for Sybase partitions .

In order to successfully convert the raw partitions into XLV, the following steps are to be strictly adhered to.

BACKUP all databases.

BCP the syslogins table from master.

bcp master..syslogins out file.out -Usa -P -c

bcp master..sysloginroles out file.out -Usa -P -c

Save all information regarding the sysdevices and database options by executing the following command:

sp_helpdb db_name -- each database

sp_helpdevice

sp_helpdb

Shutdown the backup and sql servers.

After the partitions have been updated to XLV, chown the sybase disks to sybase:users. Also, change the /etc/init.d/sybase script.

Cleanup some old sybase files from $SYBASE/devices and $SYBASE/install.

Execute sybinit to initialize a NEW sql server/backup (same server name).

Initialize sybsystemprocs database, as well.

Change sa password. (sp_password NULL,newpass)

Up the number of devices. (sp_configure "number of devices",20)

alter the database for master and tempdb.

Change sysservers to name the server, as well as the new backup server.

sp_addserver nodename_srvr,local,nodename_srvr

sp_addserver SYB_BACKUP,local,nodename_backup

execute scripts to initialize the devices and to create databases (for load).

load database dumps. Verify that database data and log are not on same device. If on same device, update sysusages by deleting the log that uses he data device and then update the size of the data device to the full size (plus the one used by the log).

uncompress the load file, manually.

online database.

BCP in the syslogins table.

BCP in the sysloginroles table.

Change the DB dbo.

Reset all DB options.

Dump all databases.

Passwords Security

Security has become a sensitive issue throughout the IT Industry. The ECS program is also concern about security and the risks associated with security. As a result the following directive is issued to all DAACs.

All System Administrators and Database Administrators at the sites are responsible for easonable security measures when installing ECS custom software. This means:

Changing the permissions of online secure files to the minimum level required .

Backing up secure file(s) to removable media (floppy or tape) and removal of secure files immediately after installation is complete.

1. The media should then be kept in a secure location.

The following file is affected as result of this requirement on the ECS program.

A. /usr/ecs/<MODE>/CUSTOM/dbms/<SUBSYSTEM>/Ec<server>SybaseLogins.sql

B. Set permissions to 711 (user read, write, execute, group and other read only)

Changing Password

One of the most common problems a DBA encounters is a user who can't connect to a SQL Server.

Changing Password Procedure

1 At the UNIX prompt, type:

isql -U*dbastudent*

then press **Return.**

The system will prompt you for a password.

2 At the **Password:** prompt, type the ***dbastudentpassword***, then press **Return.**

3 At the Sybase prompt, type the following:

sp_password *old-password, new-password*

Note: The dba (or any user with sso_role) may change another user's password with the following syntax: (this is the most common activity performed)

sp_password *sso_role password, new-password, login name*

Account Definition

In order to connect to a SQL Server a login must be created by the DBA. That login must then be assigned to particular database(s) that the user will access. All login details are stored in the syslogins table in the **master** database.

Providing access to SQL servers and their databases consists of the following steps:

- A server login account for a new user is created.
- The user is added to a database and optionally assigned to a group.
- The user or group is granted permissions on specific commands and database objects.

The procedure involves creating a script that uses three SQL stored procedures:

- **sp_addlogin -** adds a new login name to the master database but does not grant access to any user database.
- **sp_adduser -** adds access capability to the defined database(s).
- **sp_helpuser -** checks the master database for information about database user logins on the system.

By creating and storing the script file, you can easily restore all database users and their assigned databases in case of catastrophic system failure.

Create SQL Server Login Accounts

The second step in creating the fully functional database user involves the DBA assigning a SQL Server login and password for the user. The **sp_addlogin** command is used to perform this task. It will be included in the creation script as part of the procedure.

Add User to Database(s)

After the DBA determines the lognin and password for the new user, the user must be added to the databases that will be used. The **use** and **sp_adduser** commands are used to perform this task. They will be included in the creation script as part of this procedure.

Create an SQL Server Login Procedure

1 Log on to a local host.

2 Set display to current terminal by typing: **setenv DISPLAY *IPNumber*:0.0** or **setenv DISPLAY *ServerName*:0.0**, then press **Return**.

3 Log into the server which a new login procedure is to be created up by typing: **/tools/bin/ssh *ServerName***, then press **Return**.

- If you have previously set up a secure shell passphrase and executed **sshremote**, a prompt to **Enter passphrase for RSA key '<*user@localhost*>'** appears; continue with Step 4.
- If you have not previously set up a secure shell passphrase; go to Step 5.

4 If a prompt to **Enter passphrase for RSA key '<*user@localhost*>'** appears, type your ***Passphrase*** and then press the **Enter** key. Go to step 6.

5 At the **<*user@remotehost*>'s password:** prompt, type your ***Password*** and then press the **Enter** key.

6 At the UNIX prompt, type **su-sybase**, then press **Return**.

- A password prompt is displayed.

7 Enter ***SybasePassword***, then press **Return**.

- Remember that ***SybasePassword*** is case sensitive.
- You are authenticated as yourself and returned to the UNIX prompt.
- Your new home directory is **/usr/ecs/OPS/COTS/sybase** and all required environment variables have been set.

8 At the UNIX prompt, type **cd scripts/server.users**, then press **Return**.

9 At the UNIX prompt, type **cp template.sql *UserName*.sql**, then press **Return**.

- This creates a new SQL script file into which you will type the appropriate commands to create the new user login.

10 Using the text editor of your choice, edit ***UserName*.sql** (Figure 12) and make changes to information enclosed in brackets (including the brackets) as appropriate:

```
/*****************************************/
/* name: [template.sql]                  */
/* purpose:                              */
/* written:                              */
/* revised:                              */
/* reason:                               */
/*****************************************/
     sp_addlogin [login name], [password], [default database]
     go
     use [default database]
     go
     sp_adduser [login name]
     go
     /* the following is optional, by database */
     sp_changegroup [group name], [login name]
     go
     sp_helpuser
go
```

Figure 15. Sample template.sql file for new database user login..

11 After the changes have been made, save the file according to the rules of the text editor.

12 At the UNIX prompt, type:

isql -Usa -S*ServerName* -i*UserName*.sql -o*UserName*.out

then press **Return**.

- ***ServerName*** is the name of the server on which the **master** database is stored.
- ***UserName.sql*** is the name of the script file you created in step 8.
- ***UserName.out*** is the filename of the script's output for confirmation and/or troubleshooting purposes.
- The system will prompt you for a password.

13 At the **Password:** prompt, type the ***SybaseSAPassword***, then press **Return.**

14 When the UNIX prompt is again displayed the process is complete.

15 At the UNIX prompt, type **more *UserName.out*** , then press **Return**.

- This allows you to view the ***UserName.out*** file to confirm that the user login has been created or to check for user login creation errors.

Granting or Revoking Database Access Privileges

Permissions are used to control access within a database. The DBA uses the **grant** and **revoke** statements to accomplish this. There are two types of permissions within a database, **Object** and **Command**. In general, **Object** privileges control select, insert, update, delete, and execute permissions on tables, views, and stored procedures and **Command** permissions control access to the **CREATE** statements for databases, defaults, procedures, rules, tables, and views.

The syntax for the **grant** and **revoke** statements are quite similar:

grant *privilege(s)* **to** [public | name_list | role_name]

revoke *privilege(s)* **from** [public | name_list | role_name]

The privileges that may be granted or revoked include:

- select
- update
- insert
- delete
- references
- execute

Note: It is recommended that the DBA store the privilege granting and revoking commands in ".sql" files in the **$SYBASE/scripts/..** directory along with their results.

Grant or Revoke Database Access Privileges Procedure

1 Determine the privileges that you want to grant and to which user(s).

2 Log on to a local host.

3 Set display to current terminal by typing: **setenv DISPLAY *IPNumber*:0.0** or **setenv DISPLAY *ServerName*:0.0**, then press **Return**.

4 Log into the server where the user database resides by typing: **/tools/bin/ssh *ServerName***, then press **Return**.

- If you have previously set up a secure shell passphrase and executed **sshremote**, a prompt to **Enter passphrase for RSA key '<*user@localhost*>'** appears; continue with Step 5.
- If you have not previously set up a secure shell passphrase; go to Step 6.

5 If a prompt to **Enter passphrase for RSA key '<*user@localhost*>'** appears, type your ***Passphrase*** and then press the **Enter** key. Go to step 7.

6 At the **<*user@remotehost*>'s password:** prompt, type your ***Password*** and then press the **Enter** key.

7 At the UNIX prompt, type **su-sybase**, then press **Return**.

- A password prompt is displayed.

8 Enter ***SybasePassword***, then press **Return**.

- You are authenticated as yourself and returned to the UNIX prompt.
- Your new home directory is **/usr/ecs/OPS/COTS/sybase** and all required environment variables have been set.

9 At the UNIX prompt, type **isql -Usa**, then press **Return**.

10 At the **Password:** prompt, type the ***SybaseSAPassword***, then press **Return.**

- Remember the ***SybaseSAPassword*** is case sensitive.

11 If you are granting privileges, at the ISQL prompt, type **grant *Privilege* to *Login Name***, then press **Return**.

- If more than one privilege is to be granted, repeat this step on subsequent lines.

12 If you are revoking privileges, at the ISQL prompt, type **revoke *Privilege* from *Logname***, then press **Return**.

- If more than one privilege is to be revoked, repeat this step on subsequent lines.

13 When all privileges have been assigned, type **go**, then press **Return**.

System Procedures

All of the system stored procedures delivered with SQL Server 11 are located in the **sybsystemprocs** database. Previously these system procedures were stored in the **master** database. When trying to execute a stored procedure, SQL Server first "looks" in the current database, then the sybsystemprocs database, and lastly in the master. Using this method, "common" procedures can be stored by users in sybsystemprocs and used by applications.

The following is a list of some of the most frequently asked question and their system stored procedure answers. Please note that all answers are formatted with their SQL prompts included.

What are my current configuration values?

1> sp_helpconfigure

2> go

partial sample output:

Group: Backup/Recovery

Parameter Name	Default	Memory Used	Config Value	Run Value
allow remote access	1	0	1	1
print recovery information	0	0	0	0
recovery interval in minutes	5	0	5	5

Note: This command produces volumes of output, try sp_configure [parameter name]

What devices are available (defined) on my SQL Server?

1> sp_helpdevice

2> go

Note: All device logical names, physical names, virtual numbers, and sizes are reported. This output is generated from the master..sysdevices table.

What databases are defined on my SQL Server?

1> sp_helpdb

2> go

Note: Reports names, sizes, dbo, internal dbid, create date and options for each user database.

Who is connected to my SQL Server?

1> sp_who

2> go

Note: Reports spid (internal system process number, this is not the UNIX process number), status, login name, hostname, database name, etc.

What objects are defined in a particular database?

1> select name,type from pdps_TS1..sysobjects

2> go

partial sample output:

Name	Type
sysobjects	S
TrigUpdPlEsdtParmValues	TR
PlUserParameterConstraints	U

** S=system table, TR=trigger, U=user table

What columns are in a table?

```
1> sp_help PlUserParameterConstraints
2>  go
```

partial sample output:

Name	Owner	Type
PlUserParameter Constraints	dbo	user table
Data_located_on_segment	When_created	
default	Oct 21 1997 2:04 PM	

Columnname	Type	Length …
userParmConstraintsId	numeric	5
pgeId	varchar	17
pgeParameterLogicalId	int	4

.

What users are authorized in a database?

```
1>  use pdps_TS1
2>  go
3>  sp_helpuser
2> go
```

What are the names of the devices where a database has space allocated?

```
1> sp_helpdb pdps_TS1
2> go
```

Database Tuning and Performance Monitoring

Day-to-day operation of a complex database system requires the DBA to actively and continuously monitor the performance of the database servers for degradation of processing speed, available disk space, and connectivity, to name just a few of the many configurable parameters. This section discusses some of the SQL tools that are available to assist the DBA in performing system tuning and performance monitoring.

Configure SQL Server

System 11 SQL Server has about 80 configurable parameters that can be set permanently or for single runs of a program. Using the **sp_configure** command, you can display a list of the names and current values of the parameters (Figure 16). Please refer to the Sybase System 11 Technical Overview document, Chapter 6, for specific information about what each of the parameters controls. (URL http://www-esnt.sybase.com:80/css/techinfo.nsf/DocID/ID=8200-0996)

The column titled **name** is the description of the variable. The column titled **minimum** is the minimum allowable configuration setting. The column titled **maximum** is the theoretical maximum value to which the configuration option can be set. The actual maximum value is dependent on the specific platform and available resources to the SQL Server. The column titled **config_value** reflects the current system default values. The column titled **run_value** reflects the values the system is currently utilizing, which may be different from the default values.

```
name                          minimum     maximum     config_value run_value
----------------------------- ----------- ----------- ------------ -----------
recovery interval                       1       32767            0           5
allow updates                           0           1            0           0
user connections                        5  2147483647            0          25
memory                               3850  2147483647            0        5120
open databases                          5  2147483647            0          12
locks                                5000  2147483647            0        5000
open objects                          100  2147483647            0         500
procedure cache                         1          99            0          20
fill factor                             0         100            0           0
time slice                             50        1000            0         100
database size                           2       10000            0           2
tape retention                          0         365            0           0
recovery flags                          0           1            0           0
nested triggers                         0           1            1           1
devices                                 4         256            0          10
remote access                           0           1            1           1
remote logins                           0  2147483647            0          20
remote sites                            0  2147483647            0          10
remote connections                      0  2147483647            0          20
pre-read packets                        0  2147483647            0           3
upgrade version                         0  2147483647         1002        1002
default sortorder id                    0         255           50          50
default language                        0  2147483647            0           0
language in cache                       3         100            3           3
max online engines                      1          32            1           1
min online engines                      1          32            1           1
engine adjust interval                  1          32            0           0
cpu flush                               1  2147483647          200         200
i/o flush                               1  2147483647         1000        1000
default character set id                0         255            1           1
stack size                          20480  2147483647            0       28672
password expiration interval            0       32767            0           0
audit queue size                        1       65535          100         100
additional netmem                       0  2147483647            0           0
default network packet size           512      524288            0         512
maximum network packet size           512      524288            0         512
extent i/o buffers                      0  2147483647            0           0
identity burning set factor             1     9999999         5000        5000
allow sendmsg                           0           1            0           0
sendmsg starting port number            0       65535            0           0
```

Figure 16. sp_configure sample output.

In order to perform the procedure, the DBA must have obtained the following information:

- Name of server to be configured
- New values for configuration variables.

To set SQL Server configuration variables, the DBA must have **sa_role** (SQL Server) permissions. To set SQL Server configuration variables **allow updates**, **audit queue size**, **password expiration interval**, or **remote access**, **sso_role** (SQL Server) is also required.

Some parameter values take effect as soon as you reset the value. Others, which involve memory reallocation, do not change until you reset the value and then reboot SQL Server.

Configure SQL Server Procedure

---

1 Log into the server which will be reconfigured by typing: **telnet *ServerName*** or **rlogin *ServerName***, then press **Return**.

2 If a **Login**: prompt appears, log in as yourself by typing: ***YourUserID***, then press **Return**.

- A password prompt is displayed.

3 Enter ***YourPassword***, then press **Return**.

- You are authenticated as yourself and returned to the UNIX prompt.

4 Set display to current terminal by typing: **setenv DISPLAY *IPNumber*:0.0** or **setenv DISPLAY *ServerName*:0.0**, then press **Return**.

5 Begin the SQL session by typing at the UNIX prompt **isql -S*ServerName***, then press **Return**.

6 Type **sp_configure**, press **Return**, type **go**, then press **Return.**

- A list of the configurable parameters, their maximum and minimum values, the current setting and the default values is displayed.

7 After determining which parameter(s) to reset, type:

sp_configure "*ParameterName*", *NewValue*

then press **Return**; type **go**, then press **Return.**

- ***ParameterName*** is the name from the list displayed in step 6 above. Be sure to enclose the name in double quotes.
- ***NewValue*** is the numeric value that you want to assign to ***ParameterName***.

8 Repeat step 7 above for each parameter you want to reconfigure.

9 When complete, type **quit** at the ISQL prompt, then press **Return**.

- You are returned to the UNIX prompt.

This page intentionally left blank.

Database Security and Auditing

The following statements represent examples for performing security and auditing:

1 Run sybinit and install auditing.

2 Add a login for auditing:

```
sp_addlogin ssa, ssa_password, sybsecurity
use sybsecurity
sp_changedbowner ssa
sp_role “grant”, sso_role, ssa
```

3 Enable auditing:

```
sp_auditoption “enable auditing”, “on”
sp_auditlogin loginname, “cmdtext”, “on”
```

4 To Test:

```
create a table in a database with one field
grant all on the table for the loginname
log into isql using the loginname
insert a record into the table
log into isql as ssa
select * from sysaudits where loginname =
    “loginname”
```

This page intentionally left blank.

Integrity Monitoring

Integrity Monitoring

The Database Consistency Checker (**dbcc**) is a set of utility commands for checking the logical and physical consistency of a database. Use the **dbcc** commands:

- As part of regular database maintenance (periodic checks run by the System Administrator). These checks can detect, and often correct, errors before they affect a user's ability to use SQL Server.
- To determine the extent of possible damage after a system error has occurred.
- Before backing up a database.
- Because you suspect that a database is damaged. For example, if using a particular table generates the message "Table corrupt", you can use dbcc to determine if other tables in the database are also damaged.

The integrity of the internal structures of a database depends upon the System Administrator or Database Owner running database consistency checks on a regular basis. Two major functions of dbcc are:

- Checking allocation structures (the commands checkalloc, tablealloc, and indexalloc.
- Checking page linkage and data pointers at both the page level and row level (checktable and checkdb). The next section explains page and object allocation and page linkage.

dbcc is used in the database backup scripts to determine if the database is properly configured and without errors. If errors occur, the backup will not proceed and a message is sent to the DBA with notification of the problem.

This page intentionally left blank.

Troubleshooting

Diagnosing Database System Problems

Symptoms of a Damaged master Database

A damaged master database can be caused by a media failure in the area on which master is stored, or by internal corruption in the database. A damaged master database makes itself known in one or more of these ways:

- SQL Server cannot start.
- There are frequent or debilitating segmentation faults or input/output errors.
- **dbcc** (the database consistency checker) reports damage during a regularly-scheduled check of your databases.

This page intentionally left blank.

ECS Sybase Replication Server Administration Overview

Sybase Replication Server will be used for the ECS Project starting in release 6A. Implementation of Replication Server for Earth Observing System Distributed Information System (EOSDIS) Core System (ECS) will require the support of the Management Subsystem (MSS). The deployment of the Sybase Replication software and its support tools that distribute data across Distributed Active Archive Centers (DAACs) imposes additional operational requirements beyond original concept of a stand-alone DAAC operations. This new concept will allow all science users to register only once at their specified Home-DAAC and be able to request data from all DAAC sites.

In release 6A Sybase Replication Server's most basic model, the primary copy model, will be implemented. In this model, database transactions are replicated from the primary database (located at the System Monitoring Center) to a replicate database via one or more replication servers. The model assumes 'ownership' of the data by the primary database. This means, the data can only be updated by clients (local DAACs) connected to the primary database and that clients connected to the replicate database have read-only access to the data.

The transactions are continuously replicated asynchronously from the primary to the replicate database. Replication is transparent to client applications that submit database transactions to the primary database. However, this feature creates a latent period (the time it takes to propagate the transaction across the network) during which data in the primary and replicate databases are inconsistent with each other. Client programs using these databases must account for this latency.

Sybase Replication Server implements this model using replication definitions for tables at the primary database and subscriptions for tables at the replication definitions. The replication definitions specify the location of the primary data while the subscription specifies the location of the replicate data. The replication definition and subscriptions are specified at the table or stored procedure level. (only table replication is implemented for the release 6A) Replication does not require replicating all tables in a database.

The primary copy model can be viewed as a building block to create other models. The model attempts to prevent data inconsistencies that would be created by updating and replicating copies of the same data in two databases simultaneously. The model stipulates that inserts, updates, and deletes to the data can only occur at the primary database, while select statements may occur at either the primary or the replicate database. The model assumes the enforcement of data ownership using custom developed database triggers, client code, or operational procedures.

The deployment of software that distributes data across the DAACs imposes additional operational requirements beyond stand-alone DAAC operations. These enterprise level requirements conflict with the requirements for autonomous DAAC operations in two major areas: 1) capabilities that rely on database replication will be temporarily disabled between

DAACs if the databases and/or software are at different schema/drop levels; 2) administration of replicated databases requires coordination between DAACs.

This document focuses on the operations and administration of a replication system in a multi-site configuration by examining several operation scenarios involved in replication software installation and Replication Server administration. Although Sybase Replication Server supports same-site replication for warm-standby or load balancing needs, this document will focus exclusively on the issues involved in administering Sybase Replication Server for cross-site data distribution.

Cross DAAC Primary Copy Model Components

Figure 17 illustrates, and Table 9 describes, the components used in a primary copy model that uses two replication servers. This example is for illustrative purposes only. The ECS implementation will be described in the next section.

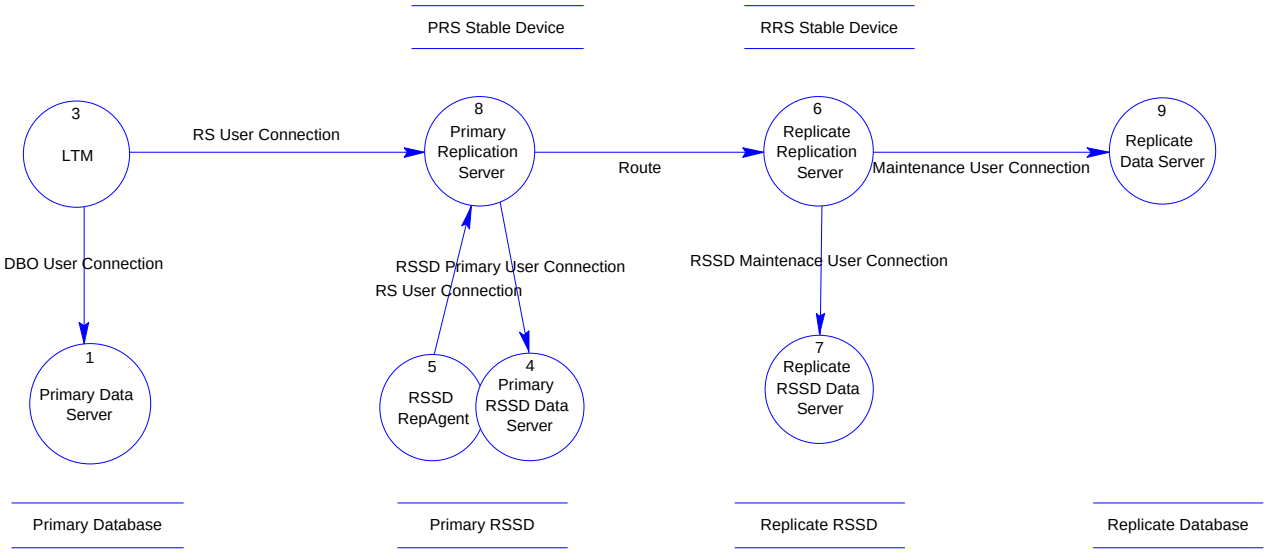


Figure 17. Replication Server Components.

Table 9. Replication Server Components.

DAAC Component	Description
Primary Data Server	The primary data server is the Sybase SQL server that maintains the primary copy of data that is being replicated.
Primary Database	Contains the copy of data that can be updated by application programs.
LTM	The log transfer manager (LTM) is a Sybase Open Server application that transfers replicate database transactions to a primary replication server and moves the secondary truncation point in the primary database transaction log. The LTM connects to the primary data server as the primary database DBO and to the primary replication server as specified when the primary database is added to the domain.
Primary Replication Server	The primary replication server (PRS) is responsible for forwarding replicate database transactions to the replicate database. The PRS maintains connections to the replicate replication servers (route) and maintains a connection to its database, the RSSD.
Primary RSSD Data Server	The primary RSSD data server maintains the primary RSSD.
RSSD RepAgent	The RSSD RepAgent is a thread in the primary RSSD data server that transfers replicate RSSD database transactions to the PRS. The RSSD RepAgent connects to the PRS as specified when the PRS is added to the domain.
Primary RSSD Database	This database houses the information required by the replication servers to operate.
PRS Stable Device	The PRS stable Device contains a FIFO queue for each primary and replicate database. Transactions are transferred from a primary database queue to a replicate database queue after the LTM sends the transaction's commit. Once a transaction is moved to the replicate database queue, the primary replication server sends the transaction to the replicate replication server.
Replicate Replication Server	The replicate replication server (RRS) is a replication server that receives replicate transactions from a primary replication server and applies the transaction to a replicate database. The RRS maintains a maintenance user connection for each replicate database.
Replicate RSSD Data Server	This server houses the RSSD for the RRS.
Replicate RSSD	This database contains information that is required for the RRS to apply replicate database transactions to a replicate database.

RRS Stable Device	The RRS stable device is a file system that contains a FIFO queue for each replicate database. Replicate database transactions are pushed into the queue before being applied to the replicate database.
ReplicateData Server	This server houses the replicate database and is updated by the RRS.
Replicate Database	The database that contains the replicate data.

Sybase Replication Server

The concept of a domain is useful when describing a replication system. Briefly, a domain is a set of replication servers and their associated components that communicate with each other. A domain can be one replication server that replicates data from a local primary database to another local replicate database (as in a warm standby application) or a domain can contain many replication servers distributed over a wide area network (WAN) as will be the case for the MSS.

Each domain requires one, and only one, ID server. An ID server is a replication server that is specified as such when it is installed. An ID Server assigns unique identifiers to domain components. The ID server must be the first replication server installed in a domain and must be accessible when any component is added to the domain.

When a replication server is installed (including the ID Server), the following components are created:

- a database called the replication server system database (RSSD) (the data server housing the RSSD must already exist)
- a stable device (queue)
- an interface (connection) to the RSSD data server
- a RepAgent for the RSSD

The RSSD contains system tables that are used by the replication server. In a mutli-server domain that implements consolidated distributed primary fragments, the RSSDs must also be replicated. The RSSD contains information about each domain component, component login ids and passwords, application specific objects such as replication definitions, replicate transaction identifiers, routes and connections, and replicate transaction errors.

The RSSD data model is documented in the manual *Replication Server Reference Manual*.

As additional replication servers are added to a domain, the replication system administrator creates replication server interfaces (RSI), or routes, between the replication servers. Routes allow replicate transactions to “flow” from a primary replication server to a replicate replication server.

Finally, application databases are added to a domain. For each database added to the domain the following components are created:

- for primary databases, an log transfer manager (LTM), which transfers database transactions from the primary database to the replication server

- for replicate database an interface from the replicate replication server to the replicate database

Replication System Administrator (RSA)

Administering the replication system is primary the role of the Replication System Administrator (RSA). The Replication System Administrator installs, configures, and administers the replication system. Given the distributed nature of the MSS implementation this role may be performed by different people at different locations. If this is the case, various tasks for administering Replication Server may require coordination between Replication System Administrators.

The Replication System Administrator has sa user permissions, which provides that person with the ability to execute nearly all commands in the replication system. In managing the system, the Replication System Administrator may need to coordinate with DBAs for both local and remote databases.

Replication System Administrators should be experienced Sybase DBAs and should have taken the Sybase training classes Replication System Administration and Replication Disaster Recovery Workshop. They should also have read and understood the manuals: *Replication Server Administration Guide*, *Replication Server Configuration Guide for UNIX Platforms*, *Replication Server Reference Manual*, and *Replication Trouble Shooting Guide*.

Replication System Administrator Task

The following tasks are required to maintain a replication system:

Table 10. Replication System Administrator Tasks (1 of 2).

Task	**Roles**
Installing Replication Server	Replication System Administrator (RSA)
Adding or removing a Replication Server	RSA
Starting up and shutting down Replication Server.	RSA
Configuring Replication Server	RSA
Maintaining Routes (Creating and modifying)	RSA
Managing the RSSD	RSA
Adding a primary and replicate database.	RSA
Adding login names, database users, and administering appropriate permissions	RSA
Adding replicated tables or changing table schemas. Creating and modifying replicated tables Creating and modifying replication definitions Creating and materializing subscriptions at replicate sites.	RSA
Defining data server function-string classes and function strings.	RSA

Table 11. Replication System Administrator Tasks (2 of 2).

Task	Roles
Applying database recovery procedures.	RSA
Maintaining and monitoring database connections	RSA
Monitoring Replication Server	RSA
Processing rejected transactions	RSA
Quiesing Replication Server	RSA
Reconciling database inconsistencies.	RSA

DAAC DBA Replication Roles and Tasks

The Database Administrator (DBA) plays a subsidiary role by supporting some Replication Server administrator task. The following are tasks that the DBA administrators will performed at the local DAACs with respect to replication server administration.

Table 12. DAAC DBA Replication Roles and Tasks.

Task	Roles
Installing Replication Server	Database Administrator (DBA)
Managing the RSSD	DBA
Adding a primary and replicate database.	DBA
Adding login names, database users, and administering appropriate permissions	DBA
Adding replicated tables or changing table schemas. Creating and modifying replicated tables Creating and modifying replication definitions Creating and materializing subscriptions at replicate sites.	DBA
Defining data server function-string classes and function strings.	DBA
Applying database recovery procedures.	DBA
Processing rejected transactions	DBA
Quiesing Replication Server	DBA
Reconciling database inconsistencies.	DBA

Sybase Replication Server Installation and Setup

Sybase Replication Server 11.5.1

The Sybase Replication server software that will be delivered to the DAACs must be installed on the MSS primary machines. Other than copying the software to the directories specified in the PSR, no further action for configuring the COTS product is necessary. This installation makes the rs_subcmp utility available to the CUSTOM scripts.

CUSTOM Installation

The replication package must be installed onto the appropriate mode/machines. Instructions will be delivered with the software drop.

The following software will need to be run:

- MSS db patch
- rs_UsrInstall

MSS database patch

Since replication will only occur between tables sharing the same schema, the MSS must be run for replication to occur successfully. The database patches will add the Sybase login mss_acct_db_maint for replication to/from the SMC in addition to bringing the database into compliance with database schema requirements.

rs_UsrInstall script

The rs_UsrInstall script will need to be configured and executed at each site. Instructions will be delivered with the software drop. The DAAC installers will need to coordinate with the SMC as to setting password, servername, and database name parameters for the replication script. This script creates the script rs_UsrMain and its associated files based are the parameters entered by the installer.

The rs_UsrMain script is the script that will need to be run for replication to occur with the SMC.

Lastly, email addresses for the replication administrators (i.e. staff who need to be notified of error conditions) will need to be added to the email notification file located in the …/CUSTOM/dbms/COM/DBAdmin directory.

Other Installation

The Sybase administrator will need to update the Sybase Interfaces files on the MSS primary servers.

Error Conditions

The output of the rs_subcmp utility is logged to EcMsRepSubCmp.log in the …/CUSTOM/logs directory. If an error condition is detected (by grepping the log after completion of the script) an email is sent to the addresses listed in the email notification file.

DAAC/SMC Coordination Issues

The DAACs should coordinate the following issues with the SMC and vice/versa.

- MSS Database Schema Versions
- Changes to the Sybase password for login mss_acct_db_maint

MSS Database Schema Version

When the SMC or a DAAC executes a database patch that changes the MSS User Profile table schema, the rs_UsrMain script will prevent execution of the rs_subcmp utility and this condition will be logged and email notification will be sent. Database patches to the MSS database should be coordinate through the SMC.

MSS login maintenance

If the password of the mss_acct_db_maint login is changed at the SMC, then the configuration files associated with rs_UsrMain will need to be updated at the SMC and at the DAACs to reflect the change.

If a DAAC changes the password of the user id, then that DAAC and SMC will need to update the configuration files associated with the rs_UsrMain script.

Replication Administration Software

Some of the Replication Server administration tasks will be supported by COTS and/or custom software (scripts). The COTS consists of the Sybase products Replication Server Manager (RSM) and Sybase Central, a GUI based administration tool.

Scripts will be developed for the following administration tasks in support of installing and configuring Replication Server and for installing replication server objects that are specific to the MSS application.

- Creating Routes
- Managing the RSSD
- Adding login names, database users, and permissions
- Creation of replication definitions, subscriptions, function strings and error classes
- Subscription materialization

Monitoring

The Sybase Central/RSM products will be used for the following tasks:

- Configuring Replication Server
- Modifying Routes
- Maintaining and monitoring database connections
- Monitoring Replication Server

Scripts that will be executed by the RSM will be developed to notify the RSA of the following events:

Component	*Event*
Servers	Active, Quiesed, Suspect, Hung, Shutdown, Dead, Unknown, Invalid
Routes	Change in status
Connection	Change in status
Partition	State change, size threshold exceeded
Queues	Latency threshold exceeded, size threshold exceeded
Database	Latency threshold exceeded

.

Recovery

Scripts will be developed to restore the RSSD or to bring application databases to a consistent state.

RSSD Recovery

- dumpdb
- dumptran
- logsegment threshold
- data segment threshold

MSS Database Recovery

- last chance logsegment threshold modification to disable secondary truncation point
- rs_subcmp scripts for each subscription in the domain

Sybase Central/RSM will be used for the following recovery tasks:

- Processing Rejected Transactions
- Quiesing Replication Server

Network and Security Requirements

The Sybase interface files used by the Replication Servers at each DAAC will need to be modified to locate all Sybase Replication and Data Server in the replication domain. Additionally, subscription materialization requires the same user id and password for the replicate replication server and the primary and replicate dataservers. Replication server userid and password maintenance must be coordinate across sites. Replication server supports password encryption, and this feature will.

A DAAC is added to the replication domain.

EROS DATA Center (EDC) is added to the domain, which before its installation includes, GSFC, LARC, and NSIDC.

Task	Role	Site
The replication server software is copied to the local host.	RSA	EDC
Replication servers and MSS SQL servers are added to the interfaces files	RSA, DBA	All
The replication server executable, and its RSSD is created.	RSA, DBA	EDC
create route SMCF to EDC	RSA	EDC
The routes are verified at each site by executing the rs_helproute command on each DAAC's RSSD ASE server.	RSA	All
The rs_init utility is executed to add EDC's mss_acct_db to the operational domain. The utility connects to the ID server at SMC to obtain unique id information for the database. The rs_init utility creates an LTM start file and starts the LTM.	RSA	EDC
Create replication definition MsAcUsrProfile	RSA	EDC
Create replication definition EcAcRequest	RSA	EDC
Create, verify, and materialized subscription MsAcUsrProfile_SMC_EDC	RSA	EDC

Fault Recovery Scenarios

General Faults

In general, Sybase Replication Server is fault tolerant. Replicate database transactions start in the primary database's transaction log, are transferred from the log to the primary replication server's queue, then to the replicate replication server's queue before being applied to the replicate databases. Database transactions are not removed from a log or a queue until the transaction has successfully moved to its next destination.

During temporary system faults, the transaction remains in its log or queue, until the fault is recovered. For example, if the replicate Sybase Server at NSIDC is shutdown for maintenance, replicate transactions from other DAACs are stored in the NSIDC's stable queue until the Sybase Server is brought back online. When NSIDC's replication server re-establishes its connection to the Sybase Server, the queued transactions will be applied to the replicate database in the order received. This approach is followed for all component failures.

When a failure occurs for an extended period or is of the type that causes a loss of replicate transactions (e.g. the failure of a devices supporting a queue or log), additional recovery steps must occur between sites.

EDC experiences an LTM failure.

The transaction log of the mss_acct_db at EDC is half full when the EDC's LTM suddenly, and expectedly, crashes. Meanwhile, a Large number of orders are requested and the database's transaction log reaches its last-chance threshold. The threshold-stored procedure fires and forces a truncation of the transaction log. The stored procedure will log an error message in the SQL Server error log to serve notice that the truncated transactions have not been replicated. The

threshold-stored procedure prevents the mss_acct_db database from 'freezing'; however, a recovery procedure will need to be used to forward the lost transactions to other DAACs.

The following tasks must occur:

Task	Role	Site
All client connections to the EDC SQL Server are suspended. Any transaction coming into EDC from other DAACs is queued in the EDC replication server stable device.	RSA, DBA	EDC
The EDC Replication Server's stable device is cleared of any open transactions.	RSA	EDC
The EDC mss_acct_db transaction log is dumped.	DBA	EDC
After verifying that EDC's transactions in the GSFC stable queue have been processed, the rs_subcmp utility is executed to update EDC's primary fragment at GSFC.	RSA	GSFC
After verifying that EDC's transactions in the LARC stable queue have been processed, the rs_subcmp utility is executed to update EDC's primary fragment at LARC.	RSA	LARC
After verifying that EDC's transactions in the NSIDC stable queue have been processed, the rs_subcmp utility is executed to update EDC's primary fragment at NSIDC.	RSA	NSIDC
The LTM at EDC is started.	RSA	EDC
Set the secondary truncation point at EDC to valid.	DBA	EDC
Resume client application connections to the EDC SQL Server.	DBA, RSA	EDC
Resume the DSI connection at EDC.	RSA	EDC

The GSFC MSS database becomes corrupt and needs to be restored from backup.

The GSFC mss_acct_db database was dumped at 12:00am. A database transaction dump executed successfully at 8:00 am. At 12:00pm, GSFC's database logs become corrupt and the SQL server takes the database off-line and suspends client connections using the database.

Task	Role	Site
The LTM is shutdown.	RSA	GSFC
Restart the GSFC Replication Server in standalone mode.	RSA	GSFC
The command admin get_generation, data_server, database is executed on the GSFC Replication Server.	RSA, DBA	GSFC
The command set log recovery for data_server.database is executed.	RSA	GSFC
A checkpoint is issued in the mss_acct_db database.	DBA	GSFC
The GSFC LTM is started with the for_recovery option.	RSA	GSFC
The 12:00 am database dump and the 8:00am transaction dump are loaded.	DBA	GSFC
The GSFC LTM is shutdown.	RSA	GSFC
The command rs_zeroltm, e0mss20_srvr, mss_acct_db is executed.	RSA	GSFC
The command dbcc settrunc('ltm', 'gen_id', <new_number>) is executed.	DBA	GSFC
The rs_subcmp utility is executed to synchronize EDC's copy of GSFC's user profile primary fragment.	RSA	EDC
The rs_subcmp utility is executed to synchronization EDC's copy of GSFC's EcAcRequest primary fragment.	RSA	EDC
The rs_subcmp utility is executed to synchronize LARC's copy of GSFC's user profile primary fragment.	RSA	LARC
The rs_subcmp utility is executed to synchronize NSIDC's copy of GSFC's user profile primary fragment.	RSA	NSIDC
Restart the GSFC Replication Server in normal mode.	RSA	GSFC
The LTM is started	RSA	GSFC
Connections are resumed at the GSFC SQL Server	DBA	GSFC

EDC RSSD becomes corrupt and needs to be restored.

RSSD recovery is different depending on the activity that occurred since the RSSD was dumped. There are four increasingly severe levels of RSSD failure with increasingly complex recovery requirements.

Activity Since Last RSSD Dump	Procedure
No DDL activity	Basic RSSD Recovery Procedure
DDL activity, but no new routes or subscriptions created	Subscription Comparison Procedure
DDL activity, no new routes created	Subscription Re-Creation Procedure
New routes created	Deintegration/Reintegration Procedure (involves removing and reinstalling replication server)

This scenario assumes that no DDL activity occurred since the last RSSD dump. DDL commands in replication command language (RCL) include those for creating, altering, or deleting routes, replication definitions, subscriptions, function strings, functions, function-string classes, or error classes.

Tasks for Basic RSSD Recovery Procedure:

Task	Role	Site
Shutdown all RepAgents and LTM that connect to the Replication Server.	RSA	EDC
Shutdown the Replication Server if it is not down.	RSA	EDC
Restore the RSSD by loading the most recent RSSD database dump and transaction dumps.	RSA, DBA	EDC
Restart the Replication Server in standalone mode.	RSA	EDC
Log into the Replication Server and get the generation number for the RSSD.	RSA	EDC
Rebuild the Replication Server queues.	RSA	EDC
Start all RepAgents and LTMs in recovery mode.	RSA	EDC
Check the loss messages in the Replication Server log, and in the logs of all Replication Servers with direct routes from the current Replication Server. (GSFC, LARC, NSIDC) If a loss is detected, see the recovery procedure for scenario The GSFC MSS database may have become corrupt and may need to be restored from backup.	RSA	All
Shutdown the LTM managed by the current Replication Server.	RSA	EDC
Execute the dbcc settrunc command at the Adaptive Server for the restored RSSD. Move up the secondary truncation point.	RSA, DBA	EDC
Execute the dbcc settrunc command at the Adaptive Server for the restored RSSD to set the generation number to one higher than the number returned in step 5.	RSA, DBA	EDC

Task	Role	Site
Restart the Replication Server in normal mode.	RSA	EDC
Restart the RepAgents for the RSSD and the LTM in normal mode.	RSA	EDC

EDC is now part of replication domain.

Reference Document

The following are Reference documents and other information that will help in the administration of Sybase Replication Server.

Name	Web site Address
Sybase Web Site	http://www.sybase.com/
Points of Contact web site Address	http://m0mss01.ecs.nasa.gov/smc/
Replication Server Reference Manual	http://www.sybase.com/products/datamove/
Sybase Central Installation Instruction	http://cmdm.east.hitc.com
Replication Sever Manager Installation Instruction	http://cmdm.east.hitc.com
609-CD-600-001, Release 6A Operations Tools Manual	http://edhs1.gsfc.nasa.gov/waisdata/catalog/rel5cat.html
Database Administrators	http://www.sybase.com
Configuration Parameter Document	http://cmdm.east.hitc.com
DBA/RSA Points of Contact at web site Address	http://m0mss01.ecs.nasa.gov/smc/
313-CD-600-001, Release 6A ECS Internal Interface Control Document for the ECS Project (5/99)	http://edhs1.gsfc.nasa.gov/waisdata/catalog/rel5cat.html
625-CD-613-001, Release 6A, ECS Project Training Material Volume 13: User Services (7/99)	http://edhs1.gsfc.nasa.gov/waisdata/catalog/rel5cat.html

Practical Exercises

Introduction

These practical exercises are presented in "day-in-the-life" scenarios relating to database administration activities. They represent real situations that you, as database administrator, are likely to encounter on a day-to-day basis.

Equipment and Materials

A functioning Release 6 system.

Starting and Stopping SQL Server

1. Check to see that the SQL Server processes are running. If they are not running, start them. If they are running, shut them down and restart them.

Database Devices

1. Create the script file(s) that will create a database device on the server to be announced from the podium. The following parameters should be met:
   - Device Name = class_device
   - Physical Name = TBA
   - Device Size = 100 MB (be sure to convert this to blocks!)
   - Virtual Device Number = you must determine this number.
2. Be sure to prepare all the appropriate files properly named and located in the correct subdirectories.
3. If time and space permit (to be determined by the instructor), perform the creation of this device.

User Databases

Margaret Janis, a data scientist, has requested that a small database be created so she can track 50 new experiments.

1. Create the script file(s) that will create a user database on the server to be announced from the podium. The following parameters should be met:
   - Database Name = mjanisdb
   - Database Device Name = class_device
   - Database Size = 50 MB
   - Transaction Log Size = 5 MB
   - Transaction Log Device = default_device
2. Be sure to prepare all the appropriate files properly named and located in the correct subdirectories.
3. If time and space permit (to be determined by the instructor), perform the creation of this database.

Backup and Recovery

1. Perform a manual backup of the autosys database. Send the backup to the autosys_backup server.
2. Be sure to prepare all the appropriate files properly named and located in the correct subdirectories.
3. Assume that the database device you created at the beginning of these exercises has failed. Perform all the steps required to recreate the database device and restore the database to full functionality.

Slide Presentation

Slide Presentation Description

The following slide presentation represents the slides used by the instructor during the conduct of this lesson.

This page intentionally left blank.